TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

MARIA SHCHERBAN

LARGE SCALE VISUALIZATION OF VIRTUAL MOVIE SETS FOR PERFORMANCE CAPTURE ACTORS

Master of Science thesis

Examiners:
Prof. Atanas Gotchev
Prof. Ossi Nykänen
Examiner and topic approved by the
Council of the Faculty of
Computing and Electrical Engineering
on 3 June 2015

ABSTRACT

MARIA SHCHERBAN: Large scale visualization of virtual movie sets for performance capture actors
Tampere University of Technology
Master of Science thesis, 68 pages
January 2016
Master's Degree Programme in Information Technology
Major: Mathematics
Examiners: Professor Atanas Gotchev, Professor Ossi Nykänen
Keywords: Motion capture studio, Spatially immersive display, Head-tracking, Non-uniform image re-sampling, Image Processing

In recent years, motion capture technology has found extensive applications in movie and video game production. In many cases, the actor is not merely overlaid into a virtual scene, but has to interact with purely virtual 3D content. Commercial motion capture systems are able to provide reasonable real-time visualization of the virtual scene to the director and cameraman (with the help of different virtual camera systems), but unfortunately, not to the actor. Therefore, a proper visualization of virtual scenes for the actor becomes an important issue.

In the thesis, the problem of providing a proper visualisation of virtual content to the actor in the motion capture studio without confining him/her with additional peripherals like VR glasses, is addressed. The main objective is to develop a projection based CAVE-like system that provides the actor a proper visualization of the virtual content. As the actor moves through the motion capture volume, the images shown on the walls are adapted to the viewer location such that the viewer receives the intended image regardless of the relative pose of the visualization surfaces.

The outcomes of the work are: a procedure for image geometry correction in a CAVE-like system, a simulation model of the CAVE-like system with the proposed geometry correction technique, a regular-to-irregular re-sampling along with adaptive anti-aliasing method, a method for re-sampling and anti-aliasing as a problem of leas-squared curve fitting with spline basis functions, and a method for performance evaluation of the proposed system, implemented in Matlab. The proposed image based rendering approach allows independence from the image source, in contrast to the existing conventional CAVE-like systems. Therefore, it is more universally applicable. The performance evaluation methods allows system testing and analysis.

PREFACE

I would like to express a deep gratitude to my supervisor, Professor Atanas Gotchev, for giving me a great opportunity to complete my thesis, for his guidance and support during the work. I would also like to thank my supervisor, Professor Ossi Nykänen, for reviewing my thesis and providing valuable comments about it. Also, I give my sincere thanks the 3D Media Group for providing a warm and kind working environment. Also, I would like to thank Olli Suominen for help and encouragement in the thesis and work related issues. I would also like to thank my family and friends for their encouragement during the entire period of my studying as a Master student at Tampere University of Technology and thesis completion.

Tampere, 10th of January, 2016

Maria Shcherban

TABLE OF CONTENTS

LIST OF ABBREVIATIONS AND SYMBOLS

$\approx$	approximately equal to
$\arctan(x)$	arctangent of x
β_k	B-spline of degree k
$\in$	belong to
ζ	breaks of a spline
K	calibration matrix
P	camera matrix
$\times$	cross product
$\cos(x)$	cosine of x
$\circledast$	discrete convolution
$\cdot$	dot product
f	focal length
$\Leftrightarrow$	Fourier transform pair
$\tilde{\mathbf{x}}$	homogeneous 3-vector representation of the 2D point
$\tilde{\mathbf{X}}$	homogeneous 4-vector representation of the 3D point
$\mathbf{x}$	inhomogeneous 2-vector representation of the 2D point
$\mathbf{X}$	inhomogeneous 3-vector representation of the 3D point
$\mapsto$	maps to
max	maximum of a set
$\mathbf{n}_i$	normal vector to the ith visualization plane
$R_X(\alpha)$	basic rotation matrix that performs rotation of vectors by an angle α about the X axis
$\mathbb{R}^N$	set of N - dimensional real numbers
$\sin(x)$	sine of x
$\mathbb{S}_{k,\mathbf{t}}$	spline space formed by a span of B-splines of order k over the knot sequence $\mathbf{t}$
$*$	tensor product of splines
$\mathbf{t}^{\mathrm{T}}$	vector transpose

API	Application Programming Interface
2D	Two-dimensional
3D	Three-dimensional
AR	Augmented Reality

CCD	Charge-Coupled Device
DoF	Degrees-of-Freedom
FIR	Finite Impulse Response
FOV	Field Of View
IIR	Infinite Impulse Response
IR	Infrared
MSE	Mean Square Error
PSNR	Peak Signal-to-Noise Ratio
SDK	Kinect Software Development Kit
VR	Virtual Reality

1. INTRODUCTION

During the past couple of decades the level of animation in movie and video games has greatly improved. The animation of humans, animals, and other virtual creatures becomes more and more realistic. These all is possible due to developments in motion capture technology [36], rapidly developing computer graphics technology, improvements in the power of computers and graphic cards. In motion capture a live motion of and object or a person is captured, digitized and mapped to a 3D virtual model that performs the same movements as the object being captured. Then the virtual model is placed into a virtual environment. When motion capture includes capturing face expressions and gentle movements, it is referred to as *performance capture*. The process of performance capture has a lot in common with the art of acting, where the actor's emotions, as well as subtle movements, play a significant role in the final result. In many cases, the actor is not merely overlaid into a virtual scene, but has to interact with purely virtual 3D content. Commercial motion capture systems are able to provide reasonable real-time visualization of the virtual scene to the director and cameraman (with the help of virtual camera system [2]), but unfortunately, not to the actor. The actor easily loses track of the location and actions of unseen virtual characters, creating a mismatch between the real and virtual 3D worlds which prompts re-shoots and manual work in post processing and decreased acting performance. The interviews for performance capture actors have shown that visualization solutions for motion capture studios provide an insufficient level of immersion with the virtual scene. The final result depends greatly on the ability of the actor to imagine the virtual scene, which becomes a serious problem when shooting is done for complex virtual scenes. Therefore, a proper visualization of virtual scenes for immersive actor feedback becomes an important issue. The problem of visualizing the virtual content for an actor regardless of his/her position and orientation within the motion capture studio has to be solved here.

One of the ways to provide visualisation of the virtual content for the actor regardless of his/her location and orientation within the motion capture studio is with the help

of head-mounted displays [52]. However, the obvious drawback of such solution is restriction of the actor in movements by the head-worn display, which decreases the level of performance. Moreover, in most cases the actor also needs to see real objects in the studio, such as props. Therefore, virtual reality solutions that substitute view to the real world, cannot be used in this case.

A visualization solution that does not imply restriction of the actor in movements with cables or head-mounted displays, is a system that involves spatially immersive displays (SIDs) [9, 35, 48, 51, 60], i.e. displays that surround the viewer. Such displays form seamless large-scale visualization surfaces. For this reason, SIDs are used in virtual reality applications [9, 51, 35] for immersive visualization of the virtual content based on the position of the viewer. Therefore, SIDs can be used for visualization of the virtual content inside of motion capture studio, in which case the walls of the motion capture studio represent the visualization surfaces, and the imagery content that needs to be visualised, is shown on the part of the visualisation surface, which corresponds to the direction in which the actor is looking at. Therefore, position and orientation of the actor's head at each moment in time should be taken into consideration, i.e., the system should provide head-tracking of the actor.

An example of utilization of SIDs of small scale is a *head-tracked display* [35], in which rendering of the virtual content shown on the display is done form the viewpoint of the head-tracked viewer. In order for the rendered content to retain the perspective of the moving viewer in front of the fixed display, a perspective projection with an asymmetric frustum [31] is used. A system, which is a composition of head-tracked displays of larger scale, is a *CAVE* - automatic virtual environment system [9]. A CAVE is a 3- or 4- sided rear-projected room with a rear-projected floor and a ceiling. A viewer inside the CAVE is head-tracked, in order to render a view-dependent perspective projection of the virtual scene on each of the surrounding surfaces (walls of the room).

Rendering engines, used in motion capture studios for generation and utilization of 3D models of the virtual scene, such as Unity [63], Autodesk Motionbuilder [65], Unreal Engine [64], utilize perspective projection with a symmetric viewing frustum [23] to map the 3D content of the virtual scene to 2D image space. Thus, rendering of the 3D virtual content with perspective projection with an asymmetric viewing frustum needs access to the rendering engine, which is a complex task. Therefore,

a solution that provides independence from the image source can simplify the task. The system that uses any image of the virtual scene and visualises it on the display surfaces without visual distortion for the viewer, meets the functionality described above. However, a mere visualisation of the imagery content on the display surfaces based on the location and orientation of the actor's head without considering the geometry of the display surfaces, will cause visual distortion of the perceived image. In order to address this issue, a content to be visualized has to be pre-distorted based on the geometry of the visualization surface.

1.1 Objectives and Scope

The aim of the work is to create a CAVE-like system that is able to make the motion capture environment more immersive by providing the actor with proper real-time visualization of virtual content. The system needs to be independent from the 3D rendering engine, i.e. any image of the virtual scene that is visualised on the display surfaces appears undistorted to the viewer. Data from the motion capture system is used to track the location of the viewer in order to render an image of the virtual scene from the viewpoint of the actor (via a 3D rendering engine by a perspective projection with symmetric viewing frustum) and then to apply a view-dependent geometry correction to the image of the virtual scene with respect to the geometry of visualization surfaces. By modelling a viewer with a pinhole camera, putting an image of the virtual scene to the image plane of the virtual camera, and mapping the 3D display surface model to a 2D model in the virtual camera plane, a view of the display surface from the virtual camera is achieved. The achieved view provides the geometry correction of the imagery content in the virtual camera plane with respect to the geometry of the visualisation surface, and after applying the calculated geometry correction, the image shown on the display looks undistorted to the real viewer. Such an image-based approach can give the desired independence of the rendering engine in terms of usage of the perspective projection. The geometry correction of the image with respect to the geometry of display surface results in the transformed image grid and the mismatch between the pixel grid of the geometry-compensated image and the pixel grid of the display, which is used for visualization. Moreover, the differences of the display resolution and the geometry corrected image resolution produce the differences in the sampling rates, which, according to the Nyquist-Shannon sampling theorem [55], result in aliasing of the visualized content. Therefore, the problem of anti-aliasing also has to be solved. An

important objective of the work is to study the options of handling these sampling related problems and their relative performance. In order to fulfil the objectives of the thesis the following tasks have to be accomplished:

- Study the pinhole camera model and perspective projection mapping
- Observe the sources of 3D content
- Study projection-based spatially immersive displays, in particular head-tracked displays and projection-based large-scale immersive displays, such as CAVE
- Study optical motion capture studios and the ways to obtain the head-tracking information about the user
- Obtain a solution for retrieving the 3D geometry of planar visualization surfaces
- Implement a procedure for image geometry correction based on the 3D geometry of visualization surfaces, by using pinhole camera model
- Study and utilize image re-sampling methods for efficient image re-sampling
- Study and utilize anti-aliasing filtering methods
- Design and implement a method to asses the performance of the image geometry correction procedure
- Implement a simulation model of the motion capture studio with a CAVE-like system for visualisation of the virtual content and asses the performance of different parts of the proposed solutions
- Implement a solution for performance evaluation of the implemented CAVE-like system

The outcomes of the thesis are the procedure for image geometry correction in a CAVE-like system, a simulation model of the CAVE-like system with the proposed geometry correction technique, a regular-to-irregular re-sampling along with adaptive anti-aliasing filtering method, a method for re-sampling and anti-aliasing as a problem of leas-squared curve fitting with spline basis functions, and a method for performance evaluation of the proposed system, implemented in Matlab. The proposed image based rendering approach allows independence from the image source,

in contrast to the existing conventional CAVE-like systems. Therefore, it is more universally applicable. The performance evaluation methods allows system testing and analysis.

1.2 Related Work

The existing visualization technique in commercial motion capture studios is made with the help of virtual camera system [2], which is used for view-dependent rendering of the pre-constructed virtual content. The rendered image is displayed either on the monitor mounted to the virtual camera rig or by the visualization display mounted to the wall of the motion capture studio. Since the display is in the fixed position, the actor easily loses track of the location and actions of unseen virtual characters, creating a mismatch between the real and virtual 3D worlds which prompts re-shoots and manual work in post processing and decreases in the acting performance.

Other possible visualisation techniques include head-mounted displays (HMDs) [52]. The obvious drawback of visualization techniques involving head-worn solutions is restriction of the actor with movements, which decreases the level of performance. Moreover, in most cases the actor also needs to see the real objects in the studio, such as props. Therefore, virtual reality solutions that substitute view to the real world, cannot be used in this case.

Different configurations of projectors available nowadays include small portable *hand-held* or *pico* projectors [57, 50], that can be held by the hands or attached to the head of the viewer for visualization of the virtual content. These solutions have similar drawbacks as other head-worn solutions described above.

The availability of different kinds of projectors enables creation of display surfaces that cover large visualization surfaces, such as seamless displays [32] and spatially immersive and semi-immersive displays [51, 9], that surround the viewer, providing a feeling of immersion with the displayed visual content. Spatially immersive displays enable creation of large scale visualization surfaces of different geometrical complexity, such as walls of the room, truncated domes or real objects. Projection of single and multiple images onto surfaces of arbitrary geometry, color and texture, covered with single or multiple projectors comprises a view-dependent geometric correction and color correction of the imagery content so that it looks undistorted

on the visualization surface. A great deal of research was carried out in this area. Some of the methods use camera assistance to find the mapping between projector and camera pixels. Then, geometry and color correction of the imagery content is done by comparing the projected and the captured images and carrying out the inverse geometry and color transformations to the projected image [20, 49]. Geometry correction methods for planar surfaces imply usage of 2D homographies [6] between projector and camera planes. Since the described methods rely on camera assisted geometry correction of the image, they are also not applicable for solving the stated problem, as they assume carrying the camera in the hands and pointing on the projection surface.

Other projection-based systems provide view-dependent stereoscopic projection in real environment [4]. The most common approach for geometric warping in this case is using the pre-calculated 3D model of the visualization surface, which can be acquired by structured light [44], depth from stereo, depth from focus methods, or combinations of them. For this a so-called *two-pass* rendering technique, described in [51], is used to render a perspectively correct imagery content. On the first pass, the view of the virtual scene from the perspective of the viewer is rendered and stored as a texture. On the next step, the rendered image is texture mapped onto the visualization surface and rendered from the viewpoint of the projector. The method for adaptation of the geometry and color of the imagery content also for dynamically changing environments is presented in [29].

The most common example of the use of spatially immersive displays is a CAVE (CAVE automatic virtual environment) [9], which is a rear-projected virtual environment, having a shape of the room, with walls, floor and ceiling used as projection surfaces, in which a user feels fully immersed with virtual environment. In this case the problem of geometry correction of the imagery content reduces to the problem of visualization on planar surfaces. The projector geometry and the geometry of projection surfaces is known a-priory. The viewer inside the room is head tracked so that the rendered image of the virtual scene retains the correct perspective. In order to render the content of the virtual scene, a perspective projection with asymmetric viewing frustum is used. A virtual camera is placed to a position of the viewer with the camera plane parallel to the projection surface [3].

The use of the CAVE-like system for motion capture studio is described in [18]. The system described there is a projection-based system which allows generation of the

3D models, as well as immersive actor feedback. The conventional rendering pipeline used in CAVE-like projection-based virtual reality systems is used there. The visualization surfaces are covered with retro-reflective cloth, in order to compensate for unevenly distributed lighting conditions.

The problem of non-uniform sampling has been heavily studied in the past years. Different possibilities of sampling exist based on the nature of the data and sampling grid, such as regular-to-regular, regular-to-irregular, irregular, listed in the increasing difficulty. The methods that address the problems of reconstruction of band limited images from irregular samples employ iterative reconstruction algorithms [1, 15] and adaptive weights [15]. The most common way for image reconstruction from irregular samples is by using splines [53, 1]. The problem of regular-to-irregular re-sampling is solved with conventional interpolation methods such as nearest-neighbour, (bi)linear, (bi)cubic, spline interpolation [45].

1.3 Thesis Structure

The thesis has the following structure. The theoretical background is given in Chapter 2. An overview of projective geometry that specifies projective camera geometry and perspective transformations is provided here. Also, 3D sensors and trackers which are used for head tracking of an actor inside the motion capture studio, and projection-based large-scale visualization techniques and sources of virtual content for them are described. The need for geometry correction of the visualised content used in conventional projection-based spatially immersive systems is then discussed. Finally, a theoretical background about image resampling and interpolation is provided, along with the overview of anti-aliasing filtering methods. Proposed solutions are described in the Chapter 3. First, the proposed solution for the head-tracked display is introduced. Then, the proposed solution for immersive visualization of the virtual content for performance actors is discussed. Here, the details of the proposed geometry correction procedure are described, the solution to handle the resampling and anti-aliasing problems is given, and the proposed solution for performance evaluation of the algorithm is presented. The results and analysis of the implemented algorithm are given in Chapter 4. The conclusions and the future work is described in Chapter 5.

2. THEORETICAL BACKGROUND

This section gives the overview of the theoretical background and state-of-the in the topic. In Section 2.1 we describe a projection principle used in a pinhole camera model, used to generate a 2D image of the 3D natural scene in CCD-like cameras. A perspective projection discussed in this section is a central projection mapping with a symmetric viewing frustum, which can be used for geometry correction of the image with respect to the 3D geometry of the visualisation surface. In Section 2.2 we present 3D scene sensors and trackers and optical motion capture, typically used nowadays for performance capturing. The 3D trackers are used to obtain a 3D tracking information about the location of an actor inside the motion capture studio, while and 3D sensors allow 3D reconstruction of visualisation surfaces. A 3D sensor which was used as a tracking sensor, namely Kinect, is also presented in this section. Section 2.3 describes projection-based large-scale visualization techniques and sources of virtual content for them. The solution for geometry correction of the visualised content used in conventional projection-based spatially immersive systems is outlined there. Section 2.4 describes the problem of interpolation in images, and discusses provides an insight on anti-aliasing filtering methods.

2.1 Projective geometry

Projective geometry is a geometry of projective transformations, that take place when points in the Euclidean 3D space are mapped to points in the Euclidean 2D space, which take place when an image is formed in the pinhole camera. In order to distort a 2D image according to the 3D geometrical structure of the visualization surfaces, the projective transformation can be used.

2.1.1 Geometric basis

2D Points. We denote 2D points in a plane $\mathbb{R}^2$ as column vectors $\mathbf{x} = [x, y]^{\mathrm{T}}$. Points can be defined by *homogeneous* coordinates. A homogeneous vector representation of a point $\mathbf{x} = [x, y]^{\mathrm{T}} \in \mathbb{R}^2$ is a 3-component vector $\tilde{\mathbf{x}} = [x_1, x_2, x_3]^{\mathrm{T}}$, where $x_3 \neq 0$. From homogeneous representation the inhomogeneous representation is achieved by $[x, y]^{\mathrm{T}} = [x_1/x_3, x_2/x_3]^{\mathrm{T}}$. We denote the inhomogeneous 2-component vector representation of a 2D-point by $\mathbf{x}$ and the homogeneous 3-component vector representation by $\tilde{\mathbf{x}}$.

3D Points. Point coordinates in 3D-space can be written in inhomogeneous coordinates as $\mathbf{X} = [X, Y, Z]^{\mathrm{T}} \in \mathbb{R}^3$ or in homogeneous form as $\tilde{\mathbf{X}} = [X_1, X_2, X_3, X_4]^{\mathrm{T}} \in \mathbb{R}^3$, where $X_4 \neq 0$. We denote the inhomogeneous 3-component vector representation of a 3D point by $\mathbf{X}$ and the homogeneous 4-component vector representation by $\tilde{\mathbf{X}}$.

2D Lines. A 2D line equation is defined as

$$ax + by + c = 0, \tag{2.1}$$

which is defined by a homogeneous 3-component vector as $\tilde{\mathbf{l}} = [a, b, c]^{\mathrm{T}}$, where $a \neq 0$ or $b \neq 0$. The equations $ax + by + c = 0$ and $(sa)x + (sb)y + sc = 0$ define the same line for any constant $s \neq 0$. Therefore, $[a, b, c]^{\mathrm{T}}$ and $s[a, b, c]^{\mathrm{T}}$ for any $s \neq 0$ are representations of the same line. The line passing through two points is defined as $\tilde{\mathbf{l}} = \tilde{\mathbf{x}} \times \tilde{\mathbf{x}}'$, where $\tilde{\mathbf{x}}$, $\tilde{\mathbf{x}}'$ are the homogeneous representations of points, and $\times$ denotes a cross product. A line $\tilde{\mathbf{l}}$ passes through a point $\tilde{\mathbf{x}}$ if $\tilde{\mathbf{x}} \cdot \tilde{\mathbf{l}} = 0$, where $\cdot$ denotes a dot product [23].

3D Planes. A 3D plane Π going through a point $\mathbf{X} = [X, Y, Z]^{\mathrm{T}}$ is defined as

$$aX + bY + cZ + d = 0, \tag{2.2}$$

where $\mathbf{n} = [a, b, c]^{\mathrm{T}}$ is the normal to the plane. The homogeneous representation of the plane is the 4-component vector $\mathbf{\Pi} = [a, b, c, d]^{\mathrm{T}}$. Consider a 3D point represented by its homogeneous coordinates, $\tilde{\mathbf{X}} = [X_1, X_2, X_3, X_4]^{\mathrm{T}}$. By taking $X = X_1/X_4$, $Y = X_2/X_4$, $Z = X_3/X_4$, the plane Equation 2.2 is

$$aX_1 + bX_2 + cX_3 + dX_4 = 0. \tag{2.3}$$

3D Lines. Any point $\mathbf{P}$ lying on a 3D line induced by two 3D points, $\boldsymbol{P_0}$, $\boldsymbol{P_1}$, is defined as follows [23, 56]

$$\mathbf{P} = \boldsymbol{P_0} + \lambda(\boldsymbol{P_1} - \boldsymbol{P_0}), \tag{2.4}$$

where λ is the segment of the line $\mathbf{P}_0\mathbf{P}_1$. Equivalently,

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} + \lambda \begin{bmatrix} X_1 - X_0 \\ Y_1 - Y_0 \\ Z_1 - Z_0 \end{bmatrix}.$$

2.1.2 Pinhole camera model

A *pinhole camera*, is an imaging device without the lens, having a shape of a box. The image of the scene is formed by the light passing through the hole and forming an inverted image on the opposite side of the box. The pinhole camera model describes 3D points are projected onto a camera's image plane. The pinhole camera geometry is depicted in Figure 2.1. Figure 2.1(a) describes the central projection

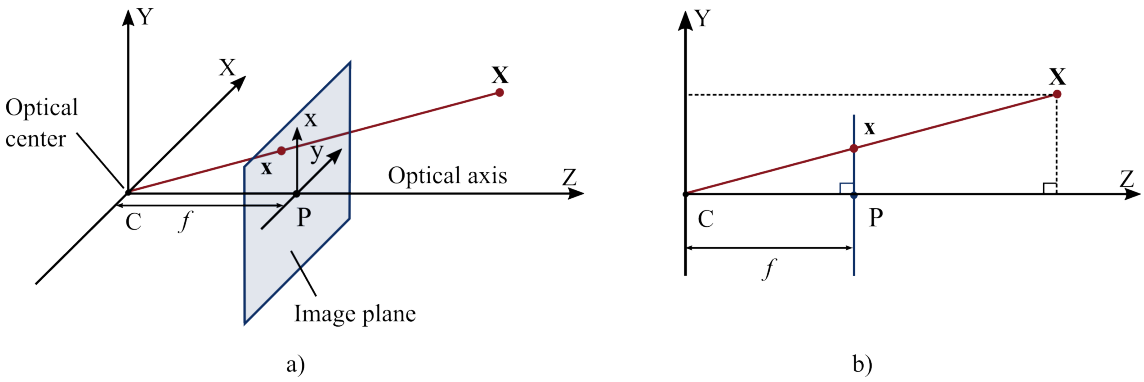

Figure 2.1 *Pinhole camera geometry. a) 3D view, b) view in YZ plane*

mapping $\mathbb{R}^3 \mapsto \mathbb{R}^2$ in Euclidean space. The *camera center* or *optical center* denoted by C, is set in the origin of the Euclidean space. Each 3D point $\mathbf{X}$ is mapped to a 2D point $\mathbf{x}$ in the *image plane* or *camera plane*. The point $\mathbf{x}$ lies in the intersection of the image plane with the line joining the optical center and the point $\mathbf{X}$. The line that goes through the optical center perpendicular to the camera plane is the *optical axis*. The point where the optical axis intersects with the camera plane is the *principal point* P. The distance between the image plane and the optical center is the *focal length* f. Figure 2.1(b) depicts the pinhole camera geometry in 2D,

in YZ plane. From similar triangles the relation between $[X, Y, Z]^{\mathrm{T}}$ and $[x, y]^{\mathrm{T}}$ is expressed as $\frac{X}{Z} = \frac{x}{f}$ and $\frac{Y}{Z} = \frac{y}{f}$. Therefore, $[x, y]^T = [fX/Z, fY/Z]^T$ and

$$[X, Y, Z]^{\mathrm{T}} \mapsto [fX/Z, fX/Z]^{\mathrm{T}} \tag{2.5}$$

is the central projection mapping $\mathbb{R}^3 \mapsto \mathbb{R}^2$ of the point in 3D-space $[X, Y, Z]^T$ to the point in 2D-space $[x, y]^T$ under the pinhole camera model [23]. The central projection mapping given by Equation 2.5 is expressed as a matrix multiplication

$$\begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{2.6}$$

which gives a central projection mapping $\mathbb{R}^3 \mapsto \mathbb{R}^2$ of the points given by their homogeneous coordinates [23]

$$[X, Y, Z, 1]^{\mathrm{T}} \mapsto [fX, fX, Z]^{\mathrm{T}}. \tag{2.7}$$

Equations 2.5 and 2.7 represent the ideal case when the principal point is in the center of the camera plane, the origins of the camera and the Euclidean coordinate systems match with each other. It is not always the case. Usually, the coordinate system of the 3D points and the coordinate system of the camera do not coincide. Therefore, the principal point offset, rotation and translation of the camera coordinate space with respect to the Euclidean space should be taken into consideration. The central projection of a 3D point $\tilde{\mathbf{X}} = [X, Y, Z, 1]^{\mathrm{T}}$ in the world coordinate space to a 2D point $\tilde{\mathbf{x}} = [x_1, x_2, x_3]^{\mathrm{T}}$ in the camera plane becomes

$$\tilde{\mathbf{x}} = KR[I \mid -\tilde{\mathbf{C}}]\tilde{\mathbf{X}}, \tag{2.8}$$

where $K = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}$ is a *camera calibration matrix* with $[c_x, c_y]^{\mathrm{T}}$ representing the coordinates of the principal point, R is a 3×3 *rotation matrix*, I is the 3×3 identity matrix and $\tilde{\mathbf{C}}$ is 3-component vector of coordinates of the camera center. The parameters f, c_x, c_y are *intrinsic parameters* of the camera. Let $\mathbf{t} = -R\tilde{\mathbf{C}}$. We denote P as a 3×4 homogeneous *camera projection matrix* with 9 degrees of

freedom (DoF):

$$P = K[R \mid \mathbf{t}]. \tag{2.9}$$

Equation 2.8 becomes

$$\tilde{\mathbf{x}} = P\tilde{\mathbf{X}}. \tag{2.10}$$

The parameters R, t are the *extrinsic parameters* of the camera. The rotation matrix R is defined as a combination of 2D rotations about each of the coordinates axes

$$R = R_x(\alpha)R_y(\beta)R_z(\gamma) \tag{2.11}$$

where

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix},$$

$$R_y(\beta) = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix},$$

$$R_z(\gamma) = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

CCD-like cameras. In CCD (charge-coupled device) like cameras, the principal point $[c_x, c_y]^T$ is expressed in pixel units. The pixels on the camera sensor are not always square thus distinguishing f_x and f_y. The camera calibration matrix K becomes

$$K = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.12}$$

where c_x, c_y, f_x, and f_y are in pixel units. Calibration matrix for such cameras has thus 11 DoF. The parameter γ is a skew-parameter, is usually set to zero in many applications, because modern sensors have perfectly aligned axes. In the pinhole camera model the points in the image plane are represented by the rays through the camera center.

2.2 3D sensors and trackers

View-dependent projection is essential to render the image of the virtual scene from the perspective of the viewer and involves head-tracking of the viewer. Also, head-tracking of the viewer is needed to locate a virtual camera within motion capture room and calculate geometrical correction of the input image of the virtual scene. These goals are achieved with the help of 3D trackers. A 3D geometry of visualization surfaces can be extracted with the help of 3D sensors. The aim of 3D tracking devices is to measure the position and orientation of an object. The object being tracked is usually marked so that the tracking device can distinguish it in 3D-space. Tracking devices are used in many areas, such as VR, augmented reality (AR), and geo-locating systems. In VR systems, the head, hands or the entire body of the person can be tracked for the purposes of virtual object manipulation, gesture-based input or viewpoint image rendering [51, 9]. In AR systems, where the synthetic objects are overlaid over the real images, trackers are used to implement the alignment of the coordinates between virtual and real objects [8]. 3D sensing technology reconstructs a visual scene in 3D space. The reconstructed information could be the 3D position of a point, or the distance to this point for example, from the center point of the depth sensing device. Depth sensors capture the depth map of the entire 3D scene within the sensor's field of view. Depth maps generated by 3D depth sensors have a form of two-dimensional (2D) images, where each pixel value corresponds to the distance to a point in a scene. Such images are referred to as *range images*, and the device used for generating those is called a *range camera*[16]. Depth maps can be either estimated by passive methods, such as depth-from-stereo [23], or sensed by active range sensors, such as structured light [44] and time of flight (ToF) [16].

Depth-from-stereo is a passive method that extracts depth information from the parallax differences between two views captured by two cameras from different viewpoints. In order to calculate the depth it is necessary to determine the corresponding points in both images. Depth calculation is then done by triangulation [23]. The cameras have to be calibrated so that relative camera positions and intrinsic parameters are known. Usually, stereo cameras are horizontally displaced from each other, and facing the same direction.

Active depth estimation methods change the scene being sensed. For example, in structured light depth estimation, a combination of camera and an illumination device, such as projector, is used. A projector illuminates the scene with a known light

pattern. The illuminated scene is captured with a camera, and depth information is determined based on analysing what geometry has to be in order to create such displacements in the image. The ToF depth estimation technique is based on measuring the time-of-flight of a light signal between the camera and the object. The light emitted from the sensor gets reflected from the surface. The distance to the object is calculated based on the phase shift.

Both the 3D trackers and depth sensors we used during completion of stated objectives. The Kinect (v1) sensor was used for depth sensing and tracking, and optical trackers used in motion capture systems for tracking. This section gives an overview about those.

2.2.1 Optical motion capture

Motion capture technology is a technology, which tracks objects in 3D-space, and captures body posture and position, and facial expressions. Motion capture is a process of sensing, digitizing and recording movements of objects or people. As a person or an object moves within the motion capture environment, the underlying motion is recorded and mapped to a digital 3D model, so that the model performs the same actions as the actor. In other words, motion capture transforms a live motion into a digital motion [36]. In a *performance capture* the face expressions, movements of fingers or other delicate expressions are also captured.

In the thesis an optical tracking system was used the position of an actor. The choice of the optical trackers was influenced by the fact that modern commercial motion capture studios commonly use optical trackers. Optical motion capture systems are widely used nowadays, since they provide accurate captured data when capturing motions. However, motion capture is not implemented in real-time, unless the motions are relatively simple [36]. Optical motion capture tracks a number of key-points (or markers) in 3D-space. These markers are placed in the moving parts of the object that best represent the movement of the part. The most efficient way is to place markers in the connections between rigid parts of the object being captured [36]. Tracking of the markers means reconstruction of their position in 3D-space within the motion capture volume in each consecutively captured frame. Optical motion capture systems typically consist of:

- Computer controller;

- Minimum of 3 light-sensitive infra-red (IR) CCD cameras;
- Passive markers.

An example of IR camera and passive markers are shown in Figure 2.2. A computer

Figure 2.2 *Componets of optical motion capture system: IR camera (left), passive markers (right).*

controls the cameras to make automatic collection of data from markers. Cameras with mounted IR lights are placed around the capture volume in order to provide capturing of the performance from different viewpoints. The frame rate of the CCD cameras has to be high enough to capture different types of motions. The frame rate of 60 to 120 frames per second is typically enough for that [36]. Cameras are calibrated so that the relative camera geometry (i.e. the intrinsic and extrinsic camera matrices) are known. Passive markers are retro-reflective balls, which are illuminated by the IR lights from the cameras and reflect the light back at the cameras, which record their 2D projections from different angles. The markers are placed on the body or object to be tracked. The captured images are put together and processed to reconstruct the 3D position of each marker.

Virtual camera. Virtual camera [2] is used in motion capture studios for the real-time visualization of the virtual scene and the virtual characters from any position and orientation within a motion capture volume. A *virtual camera rig* is a virtual camera device that can be moved by an operator within the motion capture environment to film the virtual scene and virtual characters from different perspectives. A typical virtual camera rig is depicted in Figure 2.3. The components of a virtual camera rig are a monitor, markers, and controls to start or stop recording and set camera properties. The virtual camera rig is tracked by the motion capture system, so that its position and orientation are known. The content of the virtual scene can thus be rendered from the viewpoint of the virtual camera rig.

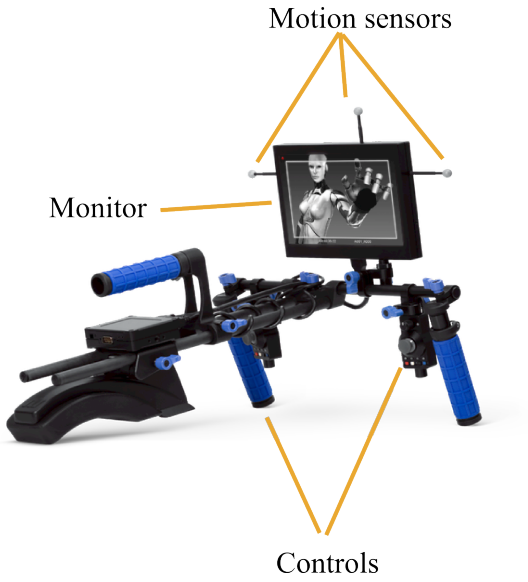

Figure 2.3 *Virtual camera rig.*

2.2.2 Structured light

Structured light technique can be used to reconstruct 3D geometry of the visualization surfaces. When the visualization surfaces are not planar, 3D geometry modelling can be achieved with the help of this method. Structured light is an active scene capture method. The principle of the method is depicted in Figure 2.4(a). One or more known light patterns are projected onto the visual scene and captured from the other viewpoint by a camera. The geometry of the scene is determined by measuring the displacements of scene content in the captured image. Typically, for a depth map estimation with structured light technique a single projector and single camera are used [25, 44]. Combinations of structured light technique with other depth acquisition methods also exist. For example, in [54] the combination of the structured lightning projector with a stereo-camera setting is observed. The method is shown to produce high accuracy depth maps. Structured light eliminates the correspondence problem that arises in passive depth-from-stereo methods. Point correspondences should not be identified here, since they are known due to projected pattern that uniquely encodes each projected pixel [47]. Coded light techniques can be classified into spatial, direct and time-multiplexing based on the type of technique that they use to label pixels. Time-multiplexing technique uses a set of projected patterns to encode pixels. Spatial encoding techniques uses a unique projected pattern, that encodes the pixel based on the neighbourhood of pixels around it. Direct

encoding techniques uses the coloured patterns that allow to encode each pixel by its grey or colour value [44].

A plenty of different structured light patterns exist, such as binary patterns [47], gray code with phase shifting patterns [21], De-Bruijn sequences [38], M-array codes [39], sinusoidal patterns [44], color patterns [7], hybrid methods [22], and even diffuse light patterns [41]. Some of these patterns are depicted in Figure 2.4(b). Figures

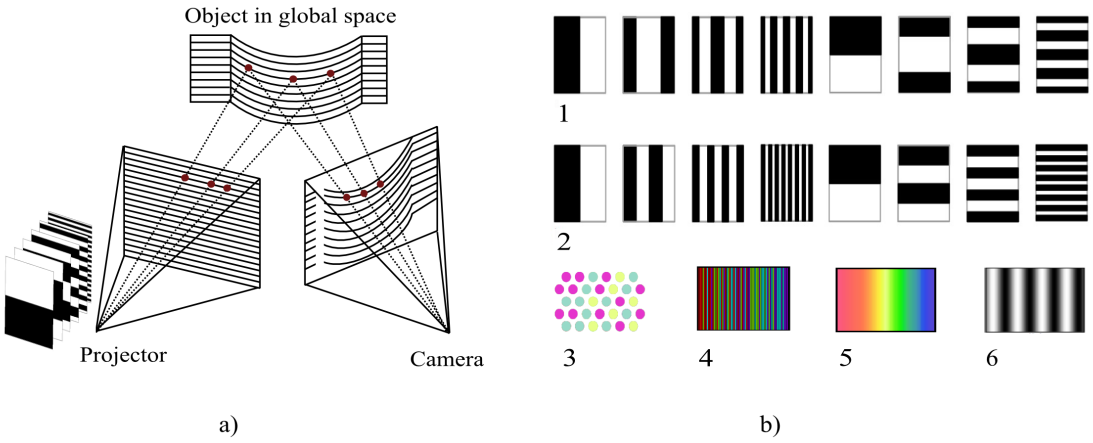

Figure 2.4 *Structured light. a)Structured light encoding principle b-1) binary-coded pattern, b-2) gray-coded pattern, b-3) M-arrays b-4) De-Bruijn sequences b-5) coloured pattern b-6) sinusoidal pattern.*

2.4(b1, b2), (b3, b4), and (b5, b6) show time-multiplexing, spatial neighbourhood, and direct encoding techniques, respectively. Time multiplexing is one of the most common structured light technique providing high accuracy depth maps. It is based on assigning a sequence of intensity values (i.e. a codeword or a label) to each coded pixel by projecting a set of known light patterns onto the scene. The most common projected pattern is a set of images that consists of stripes. These images are formed by binary coded light striping (Figure 2.4(b-1)). When the patterns are projected and captured by the camera, each pixel receives a sequence of 1s and 0s. This sequence forms a unique codeword for this pixel. In order to encode 2^m stripes a set of m projected patterns is needed. Thus, to uniquely encode a typical projector of resolution 1024×768 at most 20 patterns are projected (10 vertical and 10 horizontal). The 3D scene reconstruction is then achieved by triangulation. For that, the preliminary calibration of the camera and the projector must be done. Due to the relative shift between projector and camera, some regions of the illuminated scene cannot be captured by the camera. This regions, called *occlusions*, cannot be

reconstructed and appear in the depth map as not defined regions with no depth information. This decreases the quality of the range image.

2.2.3 Kinect (v1) sensor

Kinect sensor is a range sensing device, released by Microsoft in 2010 [61]. The Kinect sensor creates depth images of the 3D scene at a resolution 640x480. The depth map calculation is provided at a frame rate of 30 Hz. Since Kinect sensor has an affordable price, it is extensively used in various applications for both gaming and scientific purposes. The available Kinect for Windows SDK allows interaction with the sensor and provides a real-time skeleton tracking. Therefore, Kinect sensor can be used in virtual reality (VR) and augmented reality (AR) [61] applications, as well as for 3D reconstruction [28], robotics [26], navigation [43, 10, 5], motion capture and facial recognition [33], medicine [40], etc. Figure 1 shows the Kinect sensor and its main components. As shown in Figure 2.5, Kinect contains a color

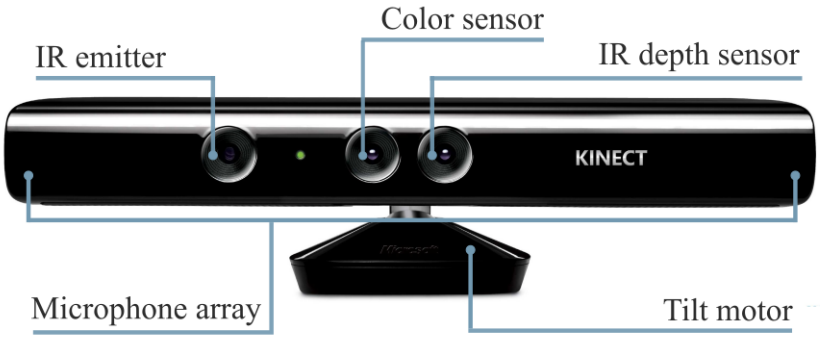

Figure 2.5 *The Kinect sensor.*

sensor, an infrared (IR) emitter, an IR depth sensor, a 4-microphone array and a tilt motor. An RGB camera captures three-channel color data at a resolution of 1280×960. A microphone array is used to sense the sound. sound. The details of the implementation of the Kinect depth sensor are not publicly available. Nevertheless, it is generally considered that it uses structured light to compute depth images of the scene. The components for depth capture are an IR emitter and an IR camera. A known speckle pattern of infrared light beams is projected onto the scene. Then, the IR camera captures the IR beams reflected back from the scene.

Since the image of the pattern is known, the speckles in the image captured by the IR camera can be matched with the speckles of the projected pattern. Due to the random nature of the projected dot pattern, the correspondence between the projected and captured dots can be determined by comparing small neighbourhoods with the help of normalized cross correlation [61]. Figure 2.6 3D reconstruction in Kinect sensor via triangulation. The planes of the IR emitter and the IR camera

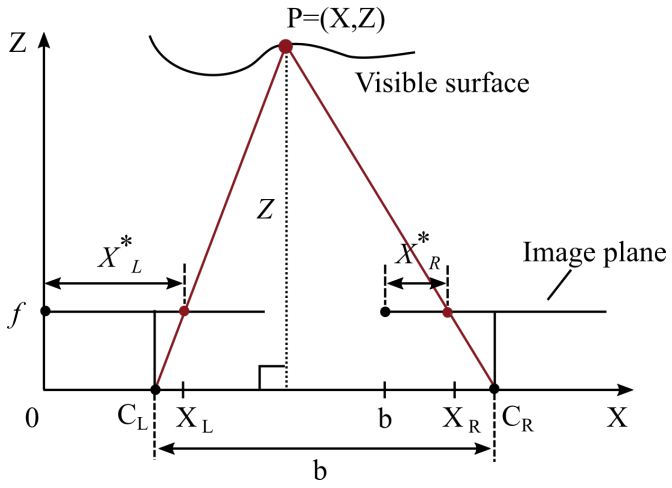

Figure 2.6 *Triangulation in Kinect depth sensor.*

are co-planar. Camera and projector are displaced only along one dimension by a distance defined by the baseline b. Also, camera and projector calibration matrices are known. The C_L, C_R are optical centers of the left projector and right camera and f is the focal length. A point P illuminated by the projector is mapped onto the IR camera plane. Image displacements in projector and camera image planes are $[X_L^*, Y_L^*]^{\mathrm{T}}$, $[X_R^*, Y_R^*]^{\mathrm{T}}$, respectively. Depth Z is calculated from similar triangles, $\frac{b}{Z} = \frac{b + X_R^* - X_L^*}{Z - f}$. Hence,

$$Z = \frac{bf}{d}, \tag{2.13}$$

where $d = X_L^* - X_R^*$ is called *disparity*. Other coordinates of the point $\mathbf{P} = [X, Y, Z]^{\mathrm{T}}$ are calculated as

$$X = \frac{X_L^* \cdot Z}{f} = b + \frac{X_R^* \cdot Z}{f}, \quad Y = \frac{X_L^* \cdot Z}{f} = \frac{X_R^* \cdot Z}{f}.$$

Since the camera and the projector are displaced only in horizontal direction, the disparity values represent the horizontal displacements. The baseline is given in meters, while the disparity and the focal length are usually given in pixel-units. The

conversion of the camera focal length to pixel units is done by $f = \frac{f_{metric}}{s}$ where s denotes the camera pixel size, f_{metric} denotes camera focal length in meters. A Kinect for Windows Software Development Kit (SDK) released in 2011 by Microsoft provides a functionality to interact with the Kinect sensor. The Microsoft for Windows SDK identifies the position and skeletal data of up to six people at a time. The skeletal data of a standing person is given by the total of 20 joints along the body. Figure 2.7 shows the set of joints that define a skeleton. In such a way Kinect sensor along with Kinect for Windows SDK can be used to track a person.

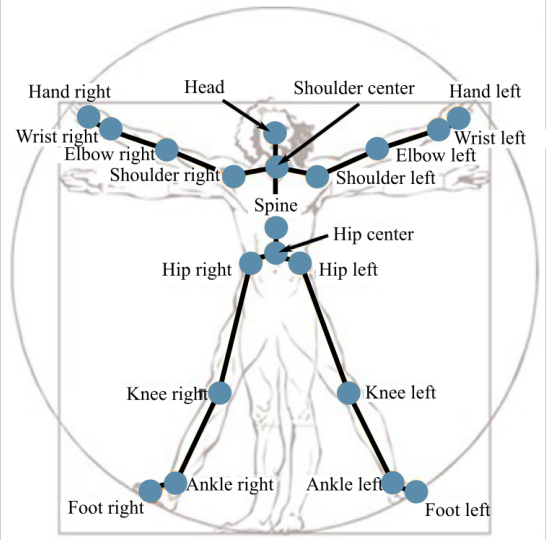

Figure 2.7 *Kinect skeletal joints.*

2.3 Large-scale visualization

In this section we describe large-scale visualization methods, namely spatially immersive displays (SIDs). Two examples of virtual reality systems, which use spatially immersive displays are given: a head-tracked display, and a CAVE - Cave Automatic Virtual Environment. These systems have the same image rendering methodology, but differ only in scale of visualisation surfaces. The choice of spatially immersive displays for large-scale visualization of the virtual content was explained in Introduction 1. Here, the aim is to see how the geometry correction of the image is done. This section also describes a two-pass rendering technique used for geometrical image correction spatially immersive displays.

2.3.1 Spatially immersive displays

A spatially immersive display (SID) is a display that surrounds the user, providing a feeling of total immersion with the displayed visual content. SIDs enable creating large scale display surfaces of planar and non-planar nature, such as as surround-screen displays, truncated domes, panoramic displays, oblique screens, transparent projection screens and even walls of the room [60, 9]. SIDs are extensively used in projection-based systems, such as VR and AR systems [9], systems for telepresence [19]. The visualisation can be done either in 2D and 3D. When augmented together, the displays form a high resolution area for visualization. A plenty of registration methods exist for registration of tiled projectors, that compose large seamless visualization surfaces of planar and non-planar nature, for example in [34]. As SIDs provide feeling of immersion with the environment, they are used in VR applications. In order to visualize 3D content and provide depth perception, a pair of stereoscopic images for each eye of the viewer is rendered and shown in a specific way. The viewer wears anaglyph or shutter glasses that separates stereoscopic images.

2.3.2 Sources of 3D content

Motion capture systems need either the 3D model of an object that is animated and a model of a 3D virtual scene. An animated 3D digital model is placed into the modelled virtual scene. The process can be visualized by the means of virtual camera rig, described in Section 2.2.1. Such content is generated by the 3D modelling software or 3D game engines.

Three-dimensional modelling software. In 3D computer graphics, 3D modelling is the process of mathematical representation of a 3D object using a software. A 3D object is mapped to 2D coordinates and displayed as a 2D image. This process is called *3D rendering*. The most commonly used 3D modelling software is provided by Autodesk (Autodesk 123D, Autodesk Maya, MotionBuilder [65]), Blender [66].

Game engine. A software framework used to create video games, is referred to as a *game engine*. Some of the components of a game engine are: a 3D rendering engine, a physics engine, a scene graph, a sound, animation, and etc. A rendering engine provides 3D rendering of 3D objects. The libraries provide access to joystick, keyboard, and mouse input devices. The most well-known game engines are Unity

and Unreal Engine. Unity and Unreal Engine are cross-platform game engines developed by Unity Technologies [63] and Epic Games [64], respectively. These game engines are used to develop video games.

2.3.3 Use of spatially immersive displays in virtual reality systems

An example of utilization of SIDs of small scale is a *head-tracked display* [35], in which rendering of the virtual content shown on the display is done form the viewpoint of the head-tracked viewer. In order for the rendered content to retain the perspective of the moving viewer in front of the fixed display, a perspective projection with an asymmetric frustum [31] is used. A system, which is a composition of head-tracked displays of larger scale is a *CAVE* - automatic virtual environment system [9] (i.e. each visualisation surface of the CAVE is a head-tracked display). CAVE is a virtual reality system, that provides a feeling of total immersion within a virtual scene. Typically, a CAVE consists of 3-, 4- rear-projected walls, a floor and a ceiling. Some of the examples of application areas for CAVE-like technologies are telepresence [19], education [27] and motion capture [18]. Figure 2.8(a) depicts the CAVE virtual reality system. Projection surfaces, projectors and a viewer are

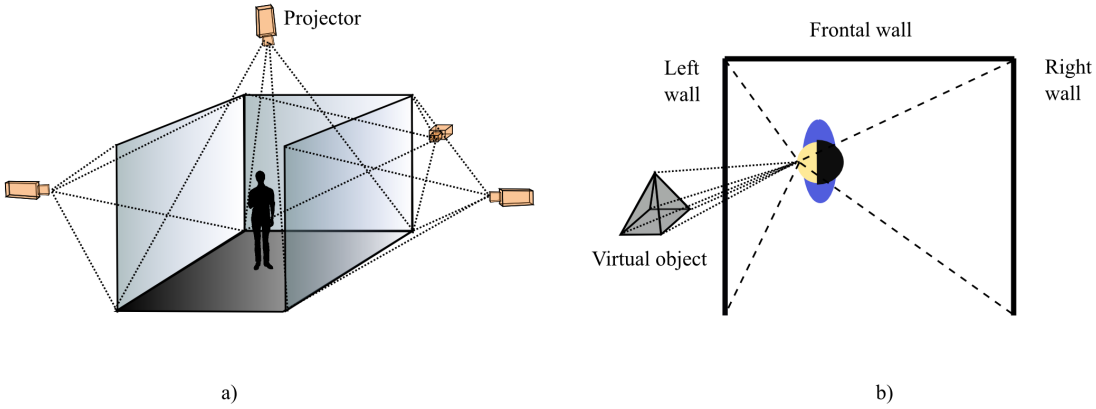

Figure 2.8 *CAVE automatic virtual reality. a) 3D view on the CAVE, b) top-down view on a moving viewer inside the CAVE. The position of the viewer defines an off-axis perspective projection onto the wall.*

located in the same world coordinate system, the 3D geometry of visualisation surfaces and projector matrices are known a-priori. The rear-projectors are calibrated

and perfectly aligned so that each projector covers one wall of the CAVE. Moreover, optical axes of projectors are orthogonal the visualisation surfaces. The area illuminated by a projector is a window to a virtual world. Models of virtual 3D objects, generated by the 3D modelling software, are placed behind the walls. A rendering approach in the CAVE-like immersive environments assumes a so-called two-pass rendering. In the first pass, an image of the 3D scene is rendered to the display domain based on the position and orientation of the head tracked viewer, and stored in the texture map. On the second pass, the texture is rendered from the viewpoint of the projector [51]. A position of the viewer with respect to the display defines the skewed (asymmetric) frustum for rendering, where the principal point does not lie in the center of the image plane. Therefore, central projection mapping cannot be used in this case. The following procedure is used in the CAVE in order to simplify geometry correction task [3]. The skewed viewing frustum d user inside the CAVE is set to match with the area of visualisation surface (i.e. a wall of the room). Therefore, the displayed image of the virtual scene, perceived "through" the projection wall, matches with the display.

2.4 Image interpolation and resampling

Resampling is one of the central problems studied in signal processing, which assumes the change of the signal sampling rate. In image processing applications resampling is often needed. For example, resampling in images is required when an image is transformed to a finer (zoom-in) or coarser (zoom-out) grid, warped based on some geometric transformation, rotated or perspectively distorted. The process of image resampling employs interpolation. Resampling of digital signals is one of the central research areas in signal processing. The theory for it can be found in signal processing books, for example [37]. First we would like to introduce the notation.

Resampling in images is the process of transforming the image from one coordinate space to another. For the simplicity, let us consider a 1-D case, for example, a row of pixels in an image, $f(u)$, $u = 0, \ldots, M-1$, $u \in Z$, where the data samples reside on a uniform integer coordinates with a unity sampling step. The discrete samples are assumed to come from some continuous function, $f_a(x)$. Resampling of the image implies 2 steps:

1. Reconstruction of the continuous signal from its discrete values.

2. Sampling the continuous signal at new sampling points.

Reconstruction of continuous signal requires approximation of the continuous function $f_a(x)$, $x \in \mathbb{R}$, by some function $g_a(x)$, based on the discrete samples of the signal $f(u)$, $u \in \mathbb{Z}$. The process of reconstruction, described above, is achieved by interpolation. For this reason, the notions of image reconstruction and interpolation are used in some sources interchangeably.

Interpolation is a process of determining the function values at the positions that lie between its samples [58]. To determine continuous function values at some intermediate positions, a continuous function is fitted through the discrete input samples. An interpolated value at some coordinate x given a discrete data $f(u)$ is defined as

$$g_a(x) = \sum_{u=\infty}^{\infty} f(u)w(x-u), \tag{2.14}$$

where $w(x)$ is a continuous convolution kernel [58]. The model defined by the equation, assumes that $f(u) = f_a(u)$. A convolution kernel must satisfy the following interpolation constraint for the case when $x = u_0$, i.e. x is an integer [58]:

$$g_a(u_0) = \sum_{x=\infty}^{\infty} f(u)w(u_0-u), \ \forall u_0 \in \mathbb{Z}, \tag{2.15}$$

which requires that interpolated values at given discrete data samples have the same value as the given data samples, i.e. $g_a(u_0) = f(u_0)$. To satisfy this condition, a continuous convolution kernel should have a a unit value in the origin and zero value at integer arguments [58]. Equation 2.15 performs an operation of discrete convolution (it operates on discrete data sequences).

A *generalized interpolation* is defined as

$$g_a(x) = \sum_{u=\infty}^{\infty} d(u)\psi(x-u), \tag{2.16}$$

for which $g_a(u_0) = f(u_0)$. In the equation $\psi(x)$ is another type of convolution kernel, called a *non-interpolating* kernel [58], for which the condition 2.15 is not specified, and $d(u)$ are the coefficients which are not restricted to be the same as $f_a(u)$. For generalized interpolation a pre-filtering operation is needed, during which the coefficients are determined. The coefficients are found by filtering operation from

the input discrete values as [58]

$$d(u) = \psi^{-1}(u) \circledast g(u), \tag{2.17}$$

where $\circledast$ denotes convolution of two discrete sequences and $\psi^{-1}(u)$ is a convolution inverse. By substituting the model coefficients in Equation 2.16 with the coefficients derived in Equation 2.17, generalized interpolation model is transferred into a classical interpolation model with an infinitely supported interpolation kernel $w(x)$.

In case of 2-D images, separable kernels are used. The interpolation kernels described above have infinite extent, which are infinite impulse response (IIR) filters. For example, an *ideal reconstruction* kernel is a *sinc* function, which has infinite extent. Nyquist-Shannon sampling criterion states that a band-limited continuous function can be recovered uniquely from its discrete samples provided that sampling rate satisfies Nyquist rate [55]. The *sinc* function is used to provide ideal reconstruction. However, *sinc* function is not a realizable in practice, since it has an infinite support. Nonetheless, there exist window methods, which consider using a truncated (windowed) version of *sinc* function [46], which are finite impulse response (FIR) filters. An example of a *window* is a rectangular window, that weights the input signal,

$$\mathrm{Rect}(x) = \begin{cases} 1 & 0 \leq |x| < 0.5 \\ 0 & 0.5 \leq |x| \end{cases}. \tag{2.18}$$

The drawback in this case is that truncation in spatial domain is done by multiplication of input signal by a rectangular window, which leads to convolution with *sinc* function is frequency domain and as a result, to ringing effect at the step edges. Other types of window function, therefore, can be used. Examples of such windows are Hanning, Hamming, Blackman, and Kaiser [46] which are non-negative smooth, bell-shaped functions. The design of IIR and FIR filters is broadly discussed in the literature about digital filter design [46].

2.4.1 Interpolation kernels

The *nearest-neighbour* is a piecewise-constant interpolation function according to which each interpolated output pixel is assigned the value of the nearest pixel in the

input image. A nearest neighbour interpolation kernel is given as

$$w(x) = \begin{cases} 1 & 0 \leq |x| < 0.5 \\ 0 & 0.5 \leq |x| \end{cases}. \tag{2.19}$$

The nearest neighbour interpolation convolution kernel is depicted in Figure 2.9.

Linear interpolation is a first order piecewise linear polynomial interpolation method that constructs a line between every two consecutive points of the input signal and each output pixel is assigned the value according to that line. All derivatives are discontinuous at the data points. Given two values $f(x_1)$, $f(x_2)$, $x_1 < x_2$, for the value x, $x \in [x_1, x_2]$, the linearly interpolated value $S(x)$ that approximates $f(x)$, $f(x) \approx S(x)$, is given by

$$S(x) = f(x_1) + (f(x_2) - f(x_1))\frac{x - x_1}{x_2 - x_1}. \tag{2.20}$$

An extension of linear interpolation for interpolation of a function of two variables, for example a 2D image, is *bilinear interpolation*. The linear interpolation is first carried out in one direction, and then in the other direction. Given the values of $f(x, y)$ at 4 image points $f(x_1, y_1)$, $f(x_2, y_1)$, $f(x_1, y_2)$ and $f(x_2, y_2)$, the bilinearly interpolated value $S(x, y)$ that approximates $f(x, y)$, $f(x, y) \approx S(x, y)$, is found by

$$\begin{aligned} S(x, y) = \frac{y_2 - y}{y_2 - y_1}\left(\frac{x_2 - x}{x_2 - x_1}f(x_1, y_1) + \frac{x - x_1}{x_2 - x_1}f(x_2, y_1)\right) \\ + \frac{y - y_1}{y_2 - y_1}\left(\frac{x_2 - x}{x_2 - x_1}f(x_1, y_2) + \frac{x - x_1}{x_2 - x_1}f(x_2, y_2)\right). \end{aligned} \tag{2.21}$$

In spatial domain, linear interpolation is equivalent to convolution of the signal with the following kernel

$$w(x) = \begin{cases} 1 - |x| & 0 \leq |x| < 1 \\ 0 & 1 \leq |x| \end{cases}. \tag{2.22}$$

The linear interpolation convolution kernel is depicted in Figure 2.9.

The *bicubic interpolation* is an extension of the cubic interpolation in a 2D regular grid. Bicubic interpolation gives smoother results than nearest neighbour or bilinear interpolation. Bicubic interpolation can be accomplished using piecewise polynomials or a cubic convolution algorithm. In order to interpolate the value of the pixel in a 2D image, a 4×4 neighbourhood is considered. The kernel is defined as follows

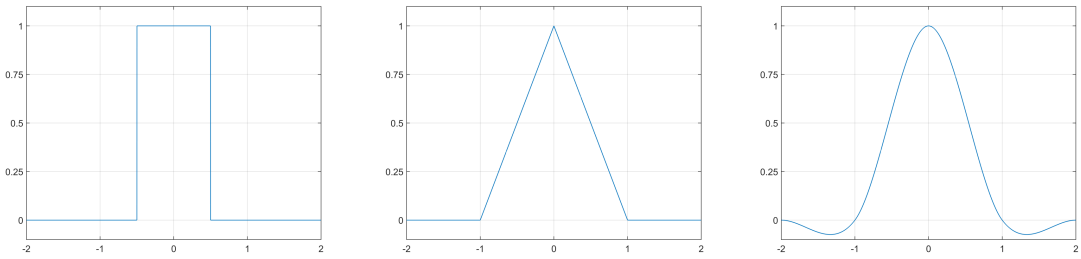

Figure 2.9 *Nearest-neighbour (left), linear (middle) and cubic (right) interpolation convolution kernels.*

[30, 45]

$$w(x) = \begin{cases} (a+2)|x|^3 - (a+3)|x|^2 + 1 & 0 \leq |x| < 1 \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a & 1 \leq |x| < 2 \\ 0 & 2 \leq |x| \end{cases}. \tag{2.23}$$

where a is usually set to $a = -0.5$ or $a = -0.75$. The convolution kernel is depicted in Figure 2.9.

2.4.2 Spline interpolation

Spline function is represented by a linear combination of B-splines $\beta_{i,k}$, $i = 1, \ldots, n$ of degree k on a non-decreasing knot sequence $t_1 \leq t_2 \leq \ldots \leq t_{n+k}$ specified by a knot vector $\mathbf{t} = [t_1, \ldots, t_{n+k}]^{\mathrm{T}}$ as [12]

$$f = \sum_{i}^{n} \alpha_i \beta_{i,k}, \;\; f \in \mathbb{S}_{k,\mathbf{t}}, \tag{2.24}$$

where α_i are unknown B-spline coefficients or *control points* and $\mathbb{S}_{k,\mathbf{t}}$ is a spline space, $\beta_{i,k}$ is zero outside the interval (t_i, t_{i+k}). $\mathbb{S}_{k,\mathbf{t}}$ is a linear space that is formed by a span of these B-splines [12]

$$\mathbb{S}_{k,\mathbf{t}} = \mathrm{span}\{\beta_{k,1}, \ldots, \beta_{k,n}\} = \{\sum_{i}^{n} \alpha_i \beta_{i,k} | \alpha_i \in \mathbb{R} \text{ for } 1 \leq i \leq n\}. \tag{2.25}$$

B-splines can be computed recursively as [12]

$$\beta_{i,1}(x) = \begin{cases} 1 & t_i \leq x \leq t_{i+1} \\ 0 & \text{otherwise} \end{cases},$$

$$\beta_{i,k}(x) = \frac{x - t_i}{t_{i+k-1} - t_i}\beta_{i,k-1}(x) + \frac{t_{i+k} - x}{t_{i+k} - t_i}\beta_{i+1,k-1}(x).$$

A spline interpolation function of order k on a given non-decreasing knot sequence $t_1 \leq t_2 \leq \ldots \leq t_{n+k}$ constructed in the way that $S(x_j) = f(x_j)$, $j = 1, \ldots, n$, can be expressed as a linear combination of B-splines of the required order k

$$S(x_j) = \sum_{i}^{n} \alpha_i \beta_{i,k}(x_j), \tag{2.26}$$

Thus, Equation 2.26 results in a set of equations, each having at most k non-zero entries [12].

2.4.3 Tensor product spline surfaces

The *tensor product* splines are used to construct bivariate spline surfaces $I = f(x, y)$. These splines are used for images, which are the functions of two variables. The tensor product of two functions $h(x)$ and $g(y)$ is a bivariate function $f(x, y) = h(x)g(y)$. Consider a spline space $\mathbb{S}_{k,\mathbf{t}}$ formed by a span of n splines of order k for a non-decreasing knot sequence $[t_1, \ldots, t_{n+k}]^{\mathrm{T}}$ and a spline $\beta_{k,i} \in \mathbb{S}_{k,\mathbf{t}}$. To construct a bivariate function, consider that coefficients of this spline are functions of y, i.e. $f(x, y) = \sum_i^n \alpha_i(y)\beta_{i,k}(x)$. Since the spline space is a span, varying y results in a spline in the same spline space $\mathbb{S}_{k,\mathbf{t}}$ on x [12]. Let the coefficient α_i be from a spline space $\mathbb{S}_{h,\mathbf{l}}$ formed by a span of m splines of order h of a non-decreasing knot sequence $\mathbf{l} = [l_1, \ldots, l_{m+h}]^{\mathrm{T}}$, i.e. $\alpha_i(y) = \sum_j^m \alpha_{ij}\beta_{j,h}(y)$, which gives a tensor product of two spline spaces

$$f(x, y) = \sum_{i=1}^{m}\sum_{j=1}^{n} \alpha_{ij}\beta_{i,k}(x)\beta_{j,h}(y), \tag{2.27}$$

in the linear spline space $\mathbb{S}_{k,\mathbf{t}} * \mathbb{S}_{h,\mathbf{l}}$, with $*$ denoting tensor product in spline space. By representing basis splines as $\mathbf{B}_n = [\beta_{1,k}, \cdots, \beta_{n,k}]^{\mathrm{T}}$ and $\mathbf{B}_m = [\beta_{1,h}, \cdots, \beta_{m,h}]^{\mathrm{T}}$ and coefficients as a matrix $A = \alpha_{ij}$, a tensor-product surface is given by a multiplication $f(x, y) = B_n(x)^{\mathrm{T}} A B_m(y)$ [12].

2.4.4 Anti-aliasing filter

In Section 2.4 we discussed resampling problem as a problem of interpolation and sampling. Also, we introduced the notions of interpolation kernels, ideal reconstruction and windowing. In the case when the sampling rate of the new samples is lower

than that of the original grid, resampling will cause aliasing. In order to prevent aliasing, an image is low-pass filtered with respect to the new sampling rate. Filtering in spatial domain is defined by convolution. The discrete convolution of two images $f(x, y)$ and $h(x, y)$ of size $M \times N$ is defined in the following way [17]

$$f(x, y) \circledast h(x, y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n)h(x - m, y - n). \tag{2.28}$$

Filter in frequency domain. As spatial convolution can be slow, faster methods are preferred. One of such methods is by filtering in frequency domain, i.e. with the help of Fourier transform. Convolution operation in spatial domain is represented by multiplication in frequency domain, $f(x, y) \circledast h(x, y) \Leftrightarrow F(u, v)H(u, v)$, where $\Leftrightarrow$ denotes transform pair. A two-dimensional discrete Fourier transform (DFT), $F(u, v)$, of an image $f(x, y)$ of size $M \times N$ and its inverse, $f(x, y)$, are defined respectively, by

$$F(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)e^{-2j\Pi(ux/M+vy/N)}, \tag{2.29}$$

$$f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v)e^{2j\Pi(ux/M+vy/N)}. \tag{2.30}$$

Filtering in frequency domain is done by transforming an image to the frequency domain, multiplying with the filter transfer function and transforming the filtered image back to the spatial domain.

3. PROPOSED SOLUTIONS

In the previous chapters we discussed the theoretical background and the state-of-the art of the projective geometry, depth sensors and trackers, large-scale visualization techniques, and sources of 3D content. This chapter is about proposed solutions to the problem of large-scale visualization of virtual sets for performance capture actors. The problem of large-scale visualization of virtual movie sets was approached in two ways, discussed in Sections 3.1 and Section 3.2.

3.1 Head-tracked display with Kinect v1 sensor

As was discussed in the previous chapters, a head-tracked display is a virtual reality system, based on spatially immersive displays. The targeted system is a projection-based CAVE-like virtual reality system, which is a system that combines several large-scale head-tracked displays (when each projection surface is represented by a head-tracked display). Therefore, one of the smaller-scale solutions for the system is a head-tracked display. Modelling such a display will help to obtain a solution for a CAVE-like system. Head-tracked display solution can help to better understand methodology and possible tools, such as tracking a viewer, perspective projection mapping, which are used later to obtain a larger-scale solution.

In this section, we observe and use a two-pass rendering technique [51], with the help of central perspective projection. In this case, the second step of the two-pass rendering technique does not need to be carried out, because the display system does not imply projector.

The main components of the system are: viewer, 3D model of the virtual scene, and a display model. A display is represented by a monitor or projector screen. To keep this solution simple, we define a 3D virtual scene ourselves. However, the solutions for rendering 3D models, described earlier, can be used (for example, Blender). The components are placed in the same coordinate system, defined by the tracking

device. In order to locate a viewer a head-tracking of the latter should be provided. A tracked data can be obtained with the help of various tracking devices, discussed in the overview of 3D trackers. Here we observe the use of Kinect sensor as a tracking device for the head-tracked display. The affordable price of Kinect sensor along with the tasks that it performs, makes it a most commonly used tool for research in VR, AR, robotics, etc. A head-tracked display with a Kinect v1 sensor is depicted in Figure 3.1. A moving viewer in front of the display is tracked by the sensor, and the

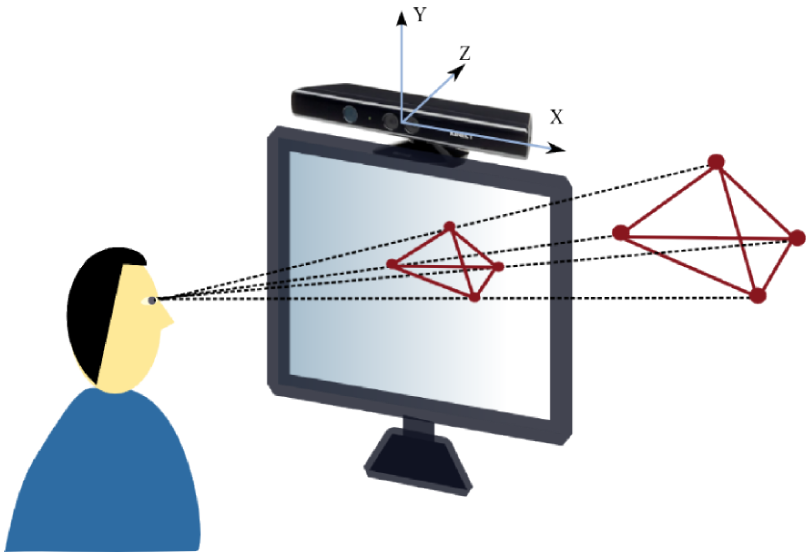

Figure 3.1 *Head-tracked display with Kinect sensor.*

display is a portal through which the viewer sees the (3D) virtual world. Virtual 3D objects, placed behind and in front of the display, are wireframe objects consisting of points and lines. The sensor defines a coordinate space for the tracked viewer and virtual objects. As in the two-pass rendering method, discussed in Section 2.3.3, first the rendering of the virtual content through the the visualisation display is done. For this, image plane is set to be parallel to the display. Figure 3.2(a) shows a top-down view on a moving viewer in front of the computer screen. It can be seen that the content has to be rendered for the position of the viewer and camera plane is parallel to the display. An off-axis projection is achieved with the help of central perspective projection.Application of the pinhole camera model for the head-tracked display is shown in Figure 3.2(b). The pinhole camera center is placed into the position of the viewer's head, denoted by C. The inhomogeneous 3-vector containing the coordinates of the head is denoted by $\tilde{\mathbf{C}}$. The image plane coincides with the computer screen, and the focal length f is given by the distance

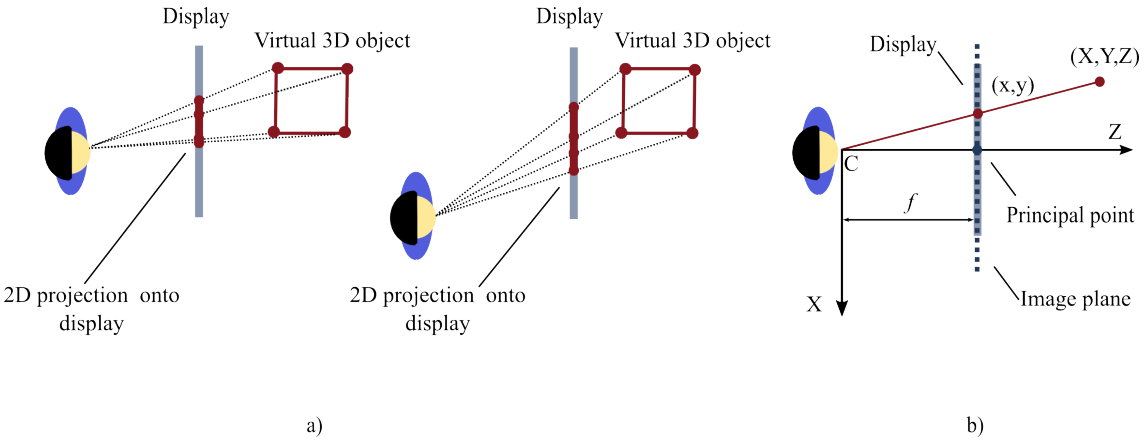

Figure 3.2 *a) Moving viewer in front of head-tracked display and content rendering, b) top-down view on the display with perspective projection parameters.*

Z between the viewer's head and the image plane and is calculated as:

$$f = \frac{Z}{s},$$

where s is a display pixel size. The coordinates of the principal point are given by the center coordinates of the computer screen, i.e. for a monitor with resolution $M \times N$ pixels, the principal point is found as $[c_x, c_y]^{\mathrm{T}} = [M/2, N/2]^{\mathrm{T}}$. In this way, each point $\tilde{\mathbf{X}} = [X, Y, Z, 1]^{\mathrm{T}}$ of the 3D object is mapped to a 2D point $\tilde{\mathbf{x}} = [x_1, x_2, x_3]$ in the image plane by Equation 2.10. The camera matrix P is given by $P = K[I \mid -\tilde{\mathbf{C}}]$, where K is a matrix of intrinsic parameters defined by Equation 2.12, and I is a rotation matrix. In this case, rotation matrix, $R = I$, because image plane is always parallel to the display. Recall that the perspective projection maps points to points and lines to lines [23]. Therefore, a line between a pair of 3D points is mapped to a line between their 2D projections. A 2D object is formed by defining a set of lines between the pairs of 3D points of the virtual object and connecting the corresponding 2D projections. Figure 3.3 shows central projection when the position of the viewer's head is not directly in the center of the display. The central projection in this case gives the wrong perspective. Here, *Projection 1* is the image of the 3D object that should be seen through the display from the viewer's position, and *Projection 2* is the rendered image with the central perspective projection setting described above. The projection of the object in the *Projection 2* is shifted, which does not give the appropriate perception of the object through the display domain. To compensate for this, the image coordinate frame has to be found. This is done by projecting

the display corner coordinates to the image plane. The mapped coordinates define the coordinate space for the projections of the virtual 3D objects. The described procedure results in off-axis perspective projection mapping of the virtual scene.

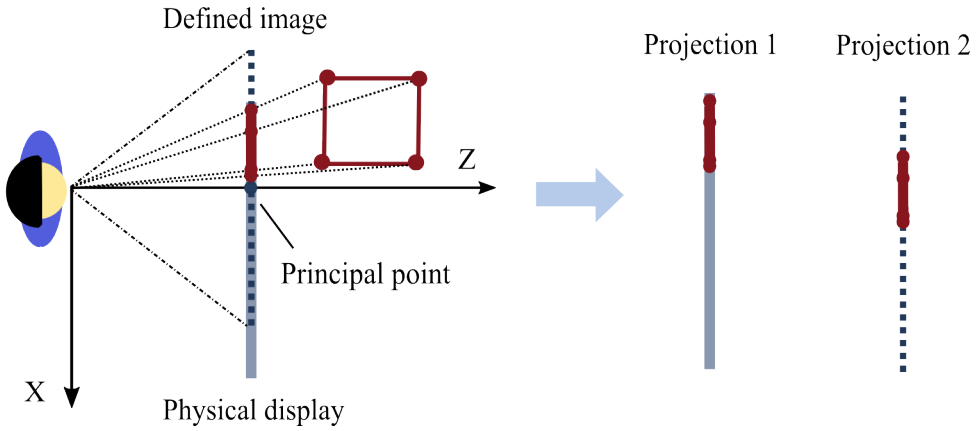

Figure 3.3 *Central perspective projection for the moving viewer with the camera plane specified by the monitor resolution and the focal length f described above.*

3.2 CAVE-like system for immersive visualization of virtual content

This section provides a description if the proposed solution for a problem of large scale visualization of virtual movie sets for performance capture actors in a CAVE-like system. The solution presented in Section 3.1 can be used for the projection based CAVE-like system. In this case, each projector which is a part of spatially immersive display is considered to be a head-tracked display. The solution presented in Section 3.1 utilises a 3D model of the virtual scene, which is needed to render a geometry corrected image of the scene. This technique is aligned with the 2-pass rendering technique. However, this solution does not provide an aimed image based rendering approach.

The proposed solution is a projection based CAVE-like system that is able to make the motion capture environment more immersive by providing the actor with proper real-time visualization of virtual content. The system combines the aspects of projection-based surround screen displays (spatially immersive displays), optical tracking and spatial augmented reality to give actors a sense of immersion with the virtual scene. The proposed CAVE-like system is shown in Figure 3.4. An

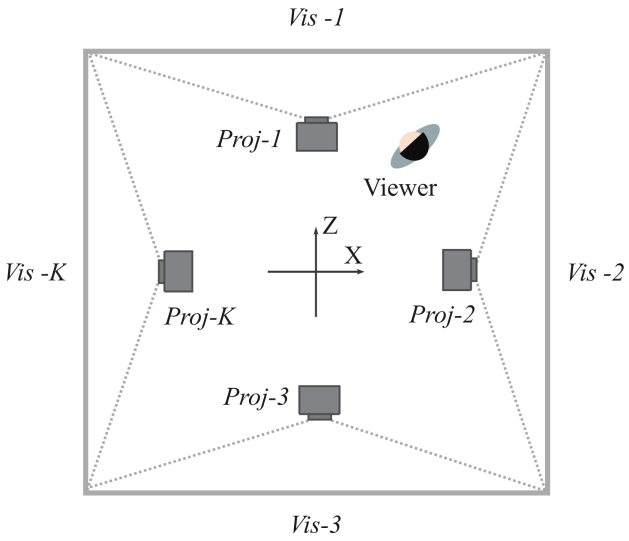

Figure 3.4 *Top-down view on the proposed system. A viewer is inside the motion capture volume, head tracked by the optical motion capture trackers. Each projector (Proj-1, ..., Proj-K) is related to a visualization surface (Vis-1, ..., Vis-K).*

actor inside the motion capture studio is tracked by the optical trackers. The main components of the system are: a tracked actor, projectors *Proj-1*, ..., *Proj-K*, and their corresponding visualization surfaces *Vis-1*, ..., *Vis-K*, with K - the number of projectors. The components of the system are placed in the world coordinate system, defined by the tracking system. The system assumes a use of a 3D rendering engine (e.g. virtual camera system used in the virtual camera rig) to render an image $f(x, y)$, $x = 0, \ldots, M_c - 1$, $y = 0, \ldots, N_c - 1$, of the virtual scene from the viewpoint of the actor, with a conventional central perspective projection mapping. The location and orientation of the actor's head is extracted from the data received from the motion capture system. As the actor moves through the volume, the images from the virtual rendering engine, shown on the walls, are geometry corrected based the the viewer location such that the viewer receives the intended image regardless of the relative pose of the visualization surfaces. For this, the viewer is modelled as a virtual pinhole camera, and the surrounding planar visualization surfaces, i.e. planar displays or projector screens, are modelled in 3D. A mapping between the input image and the addressable pixels in the environment is computed via central perspective projection. Once pre-warped through this mapping, images are displayed on the visualization surfaces and appear to the viewer as undistorted. The pre-warping requires a non-uniform re-sampling from the regular grid of the virtual camera image to the irregular grid of the projected points. A detailed description of

each main component of the proposed CAVE-like system is provided in the following sections.

3.2.1 View-dependent geometry correction

The visualization surfaces are the walls of the motion capture room, ideally covering the full field of view of the actor. The configuration can be simplified by not filling the full filed of view of the actor, which will lead to the gaps in the visualized content, depending on the location and orientation of the viewer. Projection surfaces are assumed to be planar with known dimensions. Each projector has an optical axis orthogonal to the display surface and illuminates a planar rectangular area. Regardless of the imagery content to be shown at the visualization surface, the components that define the image geometry correction are the *projector model*, the *visualization surface model* and the *location and orientation of the viewer*. The process of geometry correction of the image based on the position is shown in Figure 3.5. We put a virtual pinhole camera with to the location of the viewer, and the image from the virtual engine, $f(x, y)$, is placed to the camera plane. The geometry corrected image $\tilde{f}_i(x, y)$ shown on the ith visualization surface, $i = 1, \ldots, K$, where K is the number of projectors, i.e. the intensity value of each point on the visualization surface illuminated by the corresponding projector light ray, is formed by projection of that illuminated point to the virtual camera plane and consecutive interpolation of the color value.

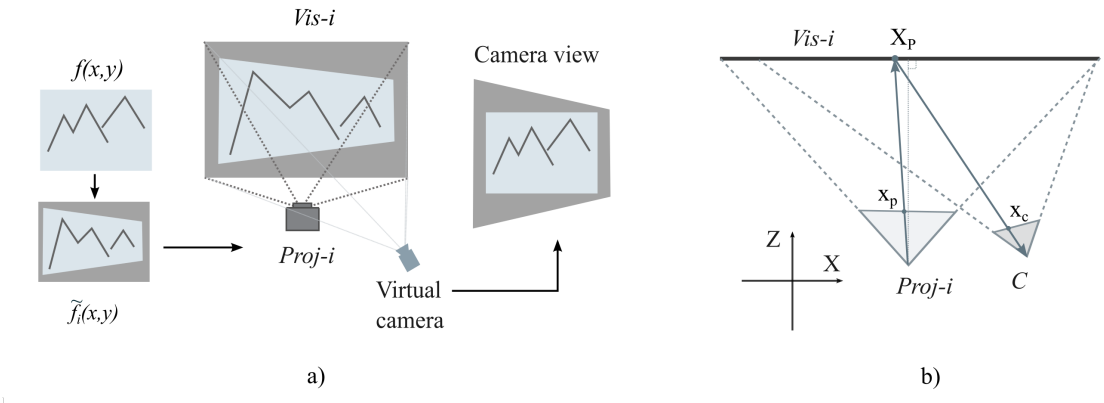

Figure 3.5 *a) Geometry correction of the image, b) top-down view on the system.*

Projector model. It is shown in [3] that projector image generation can be ap-

proximated by a pinhole projection model, similar to the pinhole camera model. The relationship between a projector 2D pixel and the corresponding 3D point illuminated on the display surface can be described using perspective 3×4 projection matrix P_p described in Section 2.1. Let the 2D pixel $\mathbf{x}_p = [x_p, y_p]^{\mathrm{T}}$ which illuminates the 3D point $\mathbf{X}_p = [X_p, Y_p, Z_p]^{\mathrm{T}}$ with the light ray coming from the projector center. The projector resolution of $M_p \times N_p$ pixels results in total of $(M_p \cdot N_p)$ points to be projected. A perspective 3D to 2D projection for the projector is defined in the similar way as the perspective projection for the pinhole camera. In homogeneous coordinates, it is given as, [3]

$$\tilde{\mathbf{x}}_p = P_p \tilde{\mathbf{X}}_p, \tag{3.1}$$

where $P_p = K_p[R_p \mid \mathbf{t}]$ is a 3×4 projection matrix and K is a projector calibration matrix, given by Equation 2.12. Since the optical axis of the projector is orthogonal to the visualization surface, the projection matrix is an identity matrix, $P_p = I$ [3]. Calculation of positions of projector's pixels on the projection surface is described further.

Display surface model. The display surface model requires information about the resolution of the projector, the corresponding size of the illuminated area and the plane equation of the visualization surface. The top-down view on the system is shown in Figure 3.6. The origin of the coordinate system defined by motion capture system is denoted as *tracker origin* and is placed in the center of the motion capture studio. The camera center is denoted by $\mathbf{C}$ defines translation $\mathbf{t}$ from the tracker origin. Since the visualization surfaces of the CAVE-like system can be modelled with planar geometry, it is possible to determine the 3D geometry of the projection surface is determined with the help of markers, rather than do it with the help of such techniques, as structured light. In order to determine the geometry from the markers, the latter are placed in the corners of each rectangle illuminated by a projector. The global coordinates of the markers are extracted with optical trackers. A 3D model of the visualization surface is then obtained by plane Equation 2.2 from at least 3 points that lie on that surface. In order to minimize possible errors caused by marker placement, we fit a 3D plane defined by Equation 2.2, $Z = aX + bY + d$, to 3D coordinates of the corners of the illuminated rectangle by least squares. A is decomposed via the singular-value decomposition $A = UDV^{\mathrm{T}}$, and the solution for $\mathbf{x} = [a, b, d]^{\mathrm{T}}$ is the last column of V. The SVD decomposition is used to minimize $||A\mathbf{x}||$ subject to $||\mathbf{x}|| = 1$.

Let the number of visualization surfaces be K. The visualization surfaces $\mathbf{\Pi}_i$ are

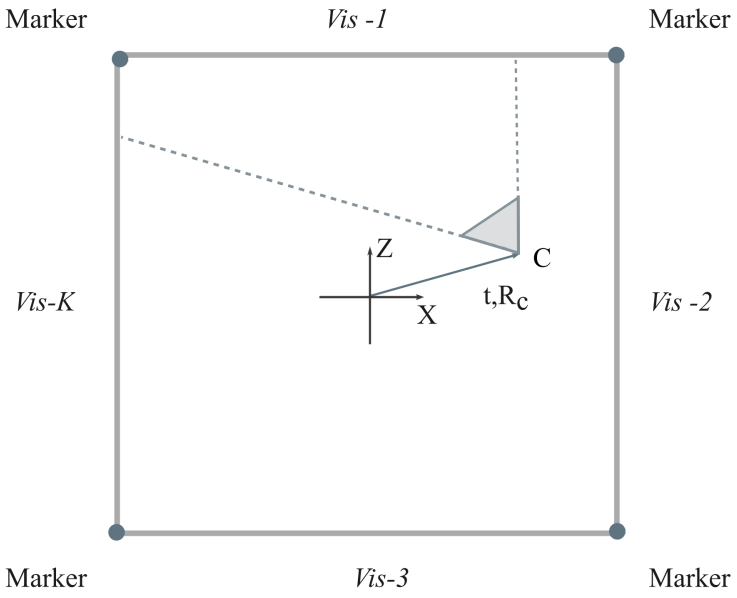

Figure 3.6 *Top-down view.*

characterized by 3×4 matrices of corner coordinates A_i, their normal vectors $\mathbf{n}_i = [a_i, b_i, c_i]^{\mathrm{T}}$, and distances from the origin of the coordinate system to the plane d_i, $i = 1, \ldots K$. For a point $\mathbf{X}_p = [X_p, Y_p, Z_p]^{\mathrm{T}}$ lying on the surface the equation of the plane $a_i X_p + b_i Y_p + c_i Z_p + d_i = 0$ becomes

$$\mathbf{n}_i \cdot \mathbf{X} + d_i = 0, \ i = 1, \ldots K, \tag{3.2}$$

where $\cdot$ is a dot product. Once the plane equation is calculated, the 3D centroids of the projector pixels are calculated. The 3D model of pixel grid is found by defining the 3D model in the origin of the global coordinate space, and then translating it to the position of the visualization surface. A 3D vector of coordinates of each 3D point $\mathbf{X}_{p_i} = [X_{p_i}, Y_{p_i}, Z_{p_i}]^{\mathrm{T}}$ illuminated by a ray through the ith projector center and 2D projector pixel $\mathbf{x}_p = [x_p, y_p]^{\mathrm{T}}$ is found as

$$\begin{bmatrix} X_{p_i} \\ Y_{p_i} \\ Z_{p_i} \end{bmatrix} = R_{p_i} \begin{bmatrix} s_{x_i}(x_{p_i} - c_{x_i}) \\ s_{y_i}(y_{p_i} - c_{y_i}) \\ 0 \end{bmatrix} + \begin{bmatrix} C_{X_i} \\ C_{Y_i} \\ C_{Z_i} \end{bmatrix}, \ i = 1, \ldots, K, \tag{3.3}$$

where R_{p_i} is a 3×3 rotation of the visualization surface with respect to the tracker coordinate space s_{x_i}, s_{y_i} are the sizes of illuminated pixels, $[c_{x_i}, c_{y_i}]^{\mathrm{T}} = [M_{p_i}/2, N_{p_i}/2]^{\mathrm{T}}$

is the principal point on the image plane of the projector and $[C_{X_i}, C_{Y_i}, C_{Z_i}]^{\mathrm{T}}$ is the principal point of the illuminated rectangle in the global space. Scale factors are found by $s_{x_i} = M^{\mathrm{m}}_{p_i}/M_{p_i}$, where $M^{\mathrm{m}}_{p_i}$ is the length of the side of the illuminated rectangle in meters. The rotation matrices of the display planes, R_{p_i}, are calculated from rotations of the normal vectors with respect to the basis vectors of the world coordinate system.

Virtual camera. Figure 3.6 shows the top-down view on the system. A tracked actor is inside of the room, where each wall comprises a visualization surface. The virtual camera plane contains the image, $f(x, y)$, $x = 0, \ldots, M_c-1, y = 0, \ldots, N_c-1$, to be visualized. The area on the visualization surface that displays imagery content is defined by the field of view of the virtual camera. The total horizontal field of view that can be seen by both human eyes is 180 degrees, 120 degrees of which form the binocular field of vision. In the vertical direction, the total field of view of the human eye is 130 degrees, which is binocular [24]. However, most of this area is out of focus. Due to the structure of human eye, the central $40-60$ degrees directly in front of the eyes perceive the most information about the scene [11]. Anything beyond this will not matter as much as quality. The choice of virtual camera parameters, i.e. camera image sensor size, focal length, field of view, is based on these considerations. For a full-frame camera sensor with physical size of 36×24 mm having an 18 mm focal length, f, the field of view

$$\alpha = 2\arctan(\frac{\text{camera sensor size}}{2f}),$$

is 90 degrees in horizontal and 60 degrees in vertical direction, which, according to the discussion above, is enough to obtain the most information about the scene. The focal length and the principal point $[c_x, c_y]^{\mathrm{T}} = [M_c/2, N_c/2]^{\mathrm{T}}$ define the intrinsic camera parameters $K_c = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$.

Viewer position $\mathbf{C}$ and orientation R_c defines the extrinsic camera parameters. Each 3D point $\mathbf{X}_{p_i} = [X_{p_i}, Y_{p_i}, Z_{p_i}]^{\mathrm{T}}$ that represents ith projector pixel is mapped to the 2D camera plane as a point $\mathbf{x}_c = [x_c, y_c]^{\mathrm{T}}$ (in homogeneous coordinates) by a central projection mapping

$$\tilde{\mathbf{x}}_c = K_c[R_c|\mathbf{t}]\tilde{\mathbf{X}}_p, \tag{3.4}$$

with $\mathbf{t} = -R_c\tilde{\mathbf{C}}$.

3.2.2 Sampling and filtering

An image to be shown by projector pixels is formed by projecting 3D projector pixels $\tilde{\mathbf{X}}_p$ onto camera plane tracked by the tracker. The 2D projections $\tilde{\mathbf{x}}_c$ land to the irregular positions within the camera plane. Depending on the relative position and orientation of the viewer and visualization surface, the resulting irregular grid of projected point has different configurations, which are shown in Figure 3.7. As

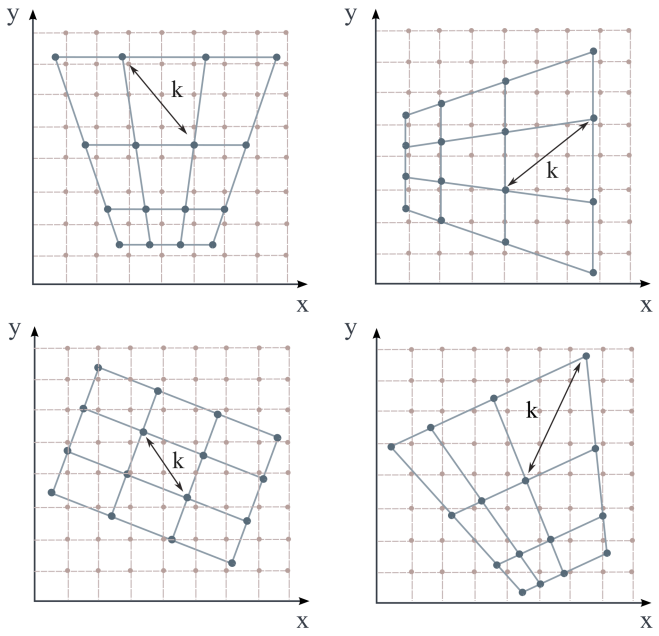

Figure 3.7 *Examples of re-sampling grid (blue line) of mapped projector 3D grid, resulted from rotations along (from left to right) top: X axis, Y axis, bottom: Z axis, X,Y,Z axis.*

can be seen in the figure, there is a substantial change in sampling rate. Therefore, a low-pass filtering is done to prevent aliasing. The cut-off frequency for the low-pass filter in this case is calculated from the worst case sampling step in the non-uniform pattern. The worst case sampling step is calculated based on the maximum Euclidean distance between closest projected grid points,

$$k = \max\{\mathrm{d}(\mathbf{x}_m, \mathbf{x}_n) \mid \mathbf{x}_m, \mathbf{x}_n \in \mathbb{R}^2\}, \tag{3.5}$$

where $m = 1, \ldots, M_{p_i}$, $n = 1, \ldots, N_{p_i}$, $i = 1, \ldots, K$ and Euclidean distance between points $\mathbf{x}_m$, $\mathbf{x}_n$ is calculated via

$$\mathrm{d}(\mathbf{x}_m, \mathbf{x}_n) = \sqrt{(x_m - x_n)^2 + (y_m - y_n)^2}. \tag{3.6}$$

Butterworth filter. We use filtering in frequency domain with a *Butterworth* filter [17]. The transfer function of the Butterworth low-pass filter of order n and cut-off frequency at a distance D_0 from the origin is given by the following formula

$$H(u,v) = \frac{1}{1 + [D(u,v)/D_0]^{2n}}, \tag{3.7}$$

where $D(u,v) = \sqrt{(u - M/2)^2 + (v - N/2)^2}$. The choice of the order of the filter effect the steepness of the transition band as well as the ringing effect of the filter [17]. The order $n = 1$ produces no ringing effect, but has a wider transition band than that of the Butterworth filter of higher orders. The cut-off frequency is calculated based on the change of the sampling rate. Suppose the sampling rate decreases k times, $k \in [0, \infty)$. Then, the maximum frequency of the signal has to be decreased k times to prevent aliasing. We calculate a cut-off frequency at a distance of

$$D_0 = \frac{M}{2k}. \tag{3.8}$$

The transfer function of the Butterwoth filter of order $n = 3$ for $k = 2.3$ is shown in Figure 3.8.

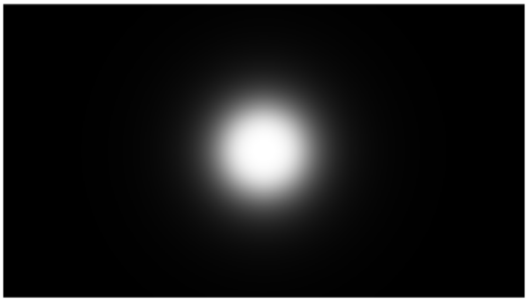

Figure 3.8 *Butterworth filter transfer function displayed as an image.*

Kaiser window filter. We use a Kaiser window, also known as the Kaiser-Bessel window [46]. Kaiser window is a one-parameter family of window functions defined

by the formula

$$w(n) = \begin{cases} \dfrac{I_0\left[\beta\sqrt{1-\left(\dfrac{n-\alpha}{\alpha}\right)^2}\right]}{I_0(\beta)}, & 0 \le n \le M\ , \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

where $\alpha = \dfrac{M}{2}$, β is a shape parameter, and I_0 is a zeroth-order modified Bessel Function of the first kind. Given filter specifications, the Kaiser window values of β, M are adjusted to achieve A as follows [46]

$$\beta = \begin{cases} 0.1102(A-8.7) & A > 50 \\ 0.5842(A-21)^{0.4} + 0.07886(A-21) & 21 \le A \le 50\ , \\ 0 & A < 21 \end{cases} \quad M \ge \frac{A-8}{2.285\Delta\omega}.$$

The cut-off frequency is calculated based on the change of the sampling rate. Suppose the sampling rate decreases k times. Then, the maximum frequency of the signal has to be decreased k times to prevent aliasing, i.e. the cut-off frequency is set as

$$f_c = \frac{1}{2k}. \tag{3.10}$$

The transfer function of the Kaiser filter of order $n = 51$, $\beta = 5$ for $k = 2.3$ is shown in Figure 3.9.

3.3 Re-sampling as a least-squares problem

Spline can be used for interpolation cases when it is required that the spline function passes exactly through the data points. Sometimes, the approximation of the data is needed, meaning that the errors between the data and the approximation spline is minimized. Such solution is given by a least-squares spline. The least squares method is commonly used to fit a surface or a curve to a given data. The spline solution for bivariate functions is a tensor-product spline. This spline can be smoothing, interpolating or approximating. The method proposed in this section uses least-squares spline to solve the problem of regular-to-irregular re-sampling. A least-squares spline is formed from the image data points, corresponding to the new sampling rate. Then, the spline is used for sampling at new grid.

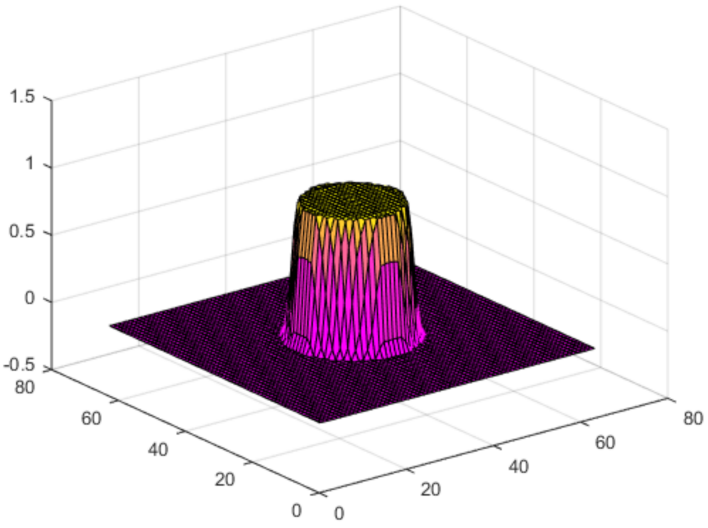

Figure 3.9 *Kaiser window filter transfer function.*

Least-squares spline for gridded data. When the spline should pass the exactly through the data points, the interpolating spline can be used. The least squares method is used to fit a surface to the data point such that the error at the data points is minimized. The tensor product spline can approximate gridded data. Consider a gridded data with known data points $f(x_i, y_j)$, $i = 1, \cdots, N$, $j = 1, \cdots, M$ the weighted least squares problem is to find an approximation $\hat{f}(x_i, y_j)$ that minimizes the residuals

$$\min_{\hat{f}(x,y)\in\mathbb{S}_{k,\mathbf{t}}*\mathbb{S}_{h,\mathbf{l}}} \sum_{i=1}^{M}\sum_{j=1}^{N} w_i v_j \left[\hat{f}(x_i, y_j) - f(x_i, y_j)\right]^2, \tag{3.11}$$

where x_i, y_j are the non-decreasing sequences of known data sites at some intervals $[a, b]$, $[c, d]$, and $w_i \geq 0$, $v_j \geq 0$ are the weights. The spline approximation surface $\hat{f}(x, y)$ is given by a tensor product, i.e. $\hat{f}(x, y) = \sum_{q=1}^{m}\sum_{p=1}^{n} \alpha_{qp}\beta_{q,k}(x)\beta_{p,h}(y)$, for the linear spline spaces $\mathbb{S}_{k,\mathbf{t}}$, $\mathbb{S}_{h,\mathbf{l}}$. and non-decreasing knot sequences $\mathbf{t} = [t_1, \ldots, l_{n+k}]^{\mathrm{T}}$, $\mathbf{l} = [l_1, \ldots, l_{m+h}]^{\mathrm{T}}$ [12]. A Spline Toolbox provided by Matlab provides least-squares spline approximation of order k for univariate and bivariate functions. In the problem of regular-to-irregular sampling and sampling rate conversion, the known image data points are the regular points of the image and the lower sampling rate defines positions of the knot sequence. The spline surface is fitted based on these assumptions in the least square sense. The spline surface is then evaluated at the irregular sites. Spline Toolbox provided by Matlab provides least-squares spline

approximation of order k for univariate and bivariate functions. In the problem of regular-to-irregular sampling and sampling rate conversion, the known data points are the regular points of the image and the lower sampling rate defines positions of the knot sequence. The spline surface is fitted based on these assumptions in the least square sense. The spline surface is then evaluated at the irregular sites. Figure 3.10 shows smoothing the 3D curve generated with a 2D image with least-squared splines. The curve on the right side of the image is a is low-passed version of the curve on the left side.

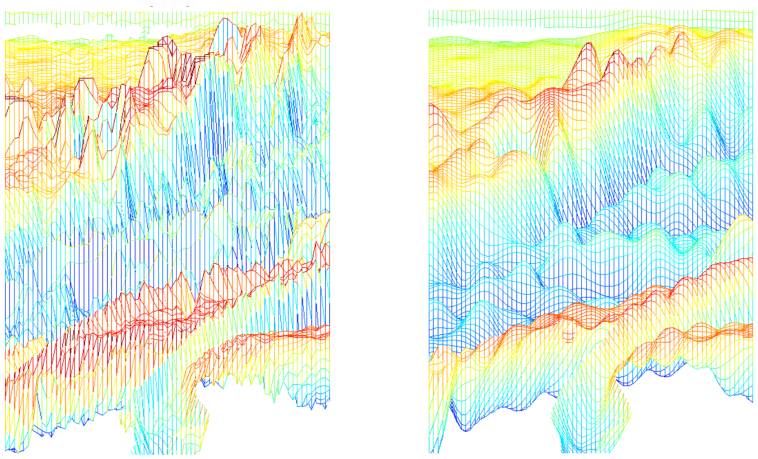

Figure 3.10 *A 3D curve (left) smoothed with least-squared splines (right), with a knot sequence corresponding to reducing the sampling rate 3 times.*

3.4 Warping procedure

The diagram of the system is shown in Figure 3.11. The image warping algorithm described previously can be summarized as follows.

Step 1. Parameters initialization

1: Calibrate motion capture cameras

2: Set the tracker coordinate space

3: **For each** projector i, $i = 1, \ldots, K$, with K- number of projectors, each associated with a visualization display

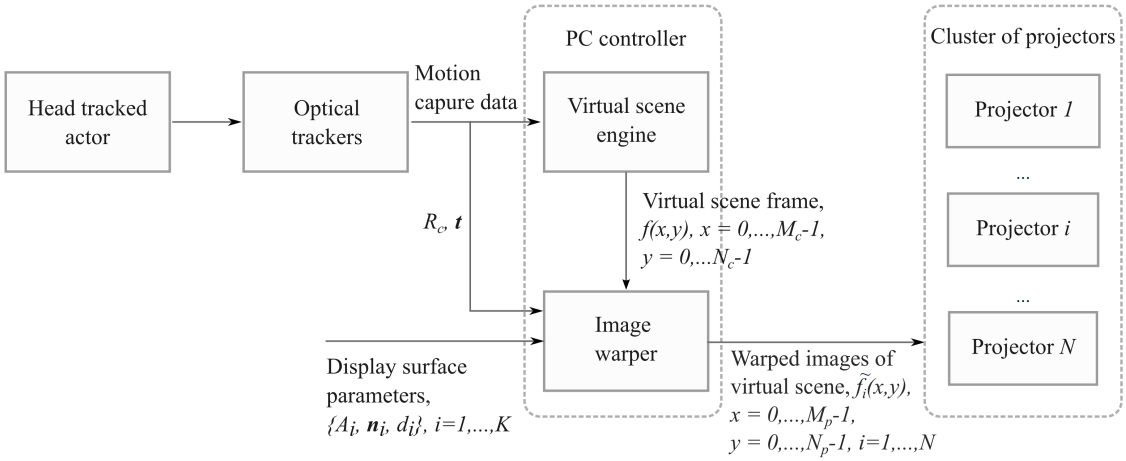

Figure 3.11 *Diagram of the system.*

4: Get corner positions A_i, with the help of markers

5: Find plane parameters $\{A_i, \mathbf{n}_i, d_i\}$, in the form $a_iX + b_iY + c_iZ + d_i = 0$, $\mathbf{n}_i = [a_i, b_i, c_i]^{\mathrm{T}}$ by applying least squares fit

6: Get rotation matrix R_{p_i}, of the projection surface

7: **For each** 2D projector pixel $\mathbf{x}_{p_i} = [x_{p_i}, y_{p_i}]^{\mathrm{T}}$

8: Get 3D pixel centroid by

$$\begin{bmatrix} X_{p_i} \\ Y_{p_i} \\ Z_{p_i} \end{bmatrix} = R_{p_i} \begin{bmatrix} s_{x_i}(x_{p_i} - c_{x_i}) \\ s_{y_i}(y_{p_i} - c_{y_i}) \\ 0 \end{bmatrix} + \begin{bmatrix} C_{X_i} \\ C_{Y_i} \\ C_{Z_i} \end{bmatrix}.$$

Step 2. Projection

1: Extract position $\mathbf{t}$ and orientation R_c of the actor from motion capture data

2: Get the image $I = f(x, y)$, of resolution $M_c \times N_c$ of the virtual scene from rendering engine

3: Place the image to the image plane of virtual camera with intrinsic parameters: focal length f, principal point $[c_x, c_y]^{\mathrm{T}} = [M_c/2, N_c/2]^{\mathrm{T}}$ and extrinsic parameters, R_c, $\mathbf{t}$

4: For each projector i, $i = 1, \ldots, K$, with K- number of projectors, each associated with a visualization display

5: For each pixel $\tilde{\mathbf{X}}_p$ in 3D projector pixel grid

6: Project 3D pixel centroid to 2D points in camera plane by $\tilde{\mathbf{x}}_c = K_c[R_c|\mathbf{t}]\tilde{\mathbf{X}}_p$

Step 3. Re-sampling

1: Get the image $I = f(x, y)$, with some resolution $M_c \times N_c$ of the virtual scene from the scene renderer

2: **For each** projector i, $i = 1, \ldots, K$, with K- number of projectors, each associated with a visualization display

3: Examine the sampling grid and find the largest sampling step k by $k = \max\{\mathrm{d}(\mathbf{x}_m, \mathbf{x}_n) \mid m = 1, \ldots, M_{p_i}, n = 1, \ldots, N_{p_i}, i = 1, \ldots, K\}$.

4: **If** $k > 1$ apply one of the following filtering and interpolation schemes to the image $I = f(x, y)$ to form the image $\tilde{f}_i(x, y)$

5: Anti-aliasing low-pass Kaiser / Gaussian filter with cut-off frequency $f_c = 1/(2k)$ with nearest / bilinear / bicubic / spline interpolation

6: Anti-aliasing low-pass Butterworth filter with cut-off frequency at distance $D_0 = M/(2k)$ with nearest / bilinear / bicubic / spline interpolation

7: Least-squares spline fit to the irregular data for anti-aliasing and sampling

8: **Else**

9: Interpolate with nearest / bilinear / bicubic / spline scheme

3.5 Mesh-based image quality evaluation method

The effectiveness of image processing algorithm can be measured by comparing the processed image with the reference image used in the algorithm, i.e. the similarity between them. A commonly used method to measure the similarity between the original image and its processed version is its peak signal-to-noise ratio (PSNR) [56]. The PSNR is usually expressed in terms of a logarithmic decibel scale. The PSNR value is derived from the average mean square error (MSE). Given a reference noise-free $M \times N$ monochrome image $f(x, y)$ and its noisy approximation $\hat{f}(x, y)$, $x = 0, \ldots, M-1$, $y = 0, \ldots, N-1$, MSE is defined as:

$$MSE = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[f(x, y) - \hat{f}(x, y) \right]^2 . \tag{3.12}$$

The PSNR is calculated as:

$$PSNR = 10 \log_{10} \left(\frac{MAX_f^2}{MSE} \right) , \tag{3.13}$$

where MAX_f is the maximum possible intensity value of the reference image $f(x, y)$, i.e. when 8 bits per pixel value is used, $MAX_f = 255$. If done on a real system, the effects like camera response function, lens distortion, and tracking error, make pixel-wise alignment required by a metrics like PSNR not accurate. Since PSNR is a pixel-wise metric, even small misalignments between the reference and the processed images will negatively impact on the results. Therefore, we propose a mesh based simulation model. Formation of the image by a projector, which optical axis is orthogonal to the visualization surface is approximated by piece-wise constant functions, i.e. rectangular projector pixels. These piece-wise constant functions form a polygonal mesh with each polygon corresponding to a pixel in the surface. Mesh formation principle is visualised in Figure 3.12. Each pixel is modelled by 3D vertices placed in the corner places of each pixel rectangle. A mesh is represented by a set of vertices, and the set of polygons, with each polygons having the color of the corresponding 3D pixel. The same central perspective projection described in Section 3.2.1 is used to project the mesh in the display domain to the virtual camera plane, located at the desired viewer position. Then, the mesh is rasterized at high resolution. Forming a mesh out of the reference image $f(x, y)$ in the virtual camera plane and rasterizing it in high resolution can cause spatial domain artefacts in the reference image. Therefore, the reference image $f(x, y)$ in the virtual camera plane

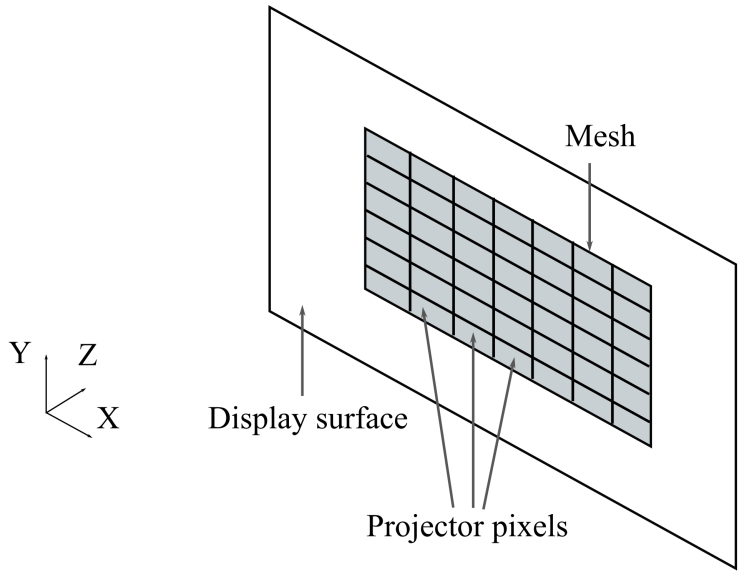

Figure 3.12 *Mesh formation principle in 3D view.*

is up-sampled by an integer factor with a nearest neighbour interpolation, and the rasterization factor of the projected mesh is chosen to provide the same resolution as the up-sampled reference image. In this way, the effect of the perspective distortion is modelled, but images can be perfectly aligned and e.g. PSNR can be computed. The block diagram of the evaluation method is given in Figure 3.13. Figure 3.13 shows how geometry corrected images $\tilde{f}_i(x, y)$, $i = 1, \ldots K$, formed for each display surface based on the position and orientation of the viewer inside the room, contribute to the final image formed at the virtual camera plane. Each mesh of geometry corrected image $f_i(x, y)$ is projected to the camera plane and contributes the final mesh by forming a part of the reference image $f(x, y)$.

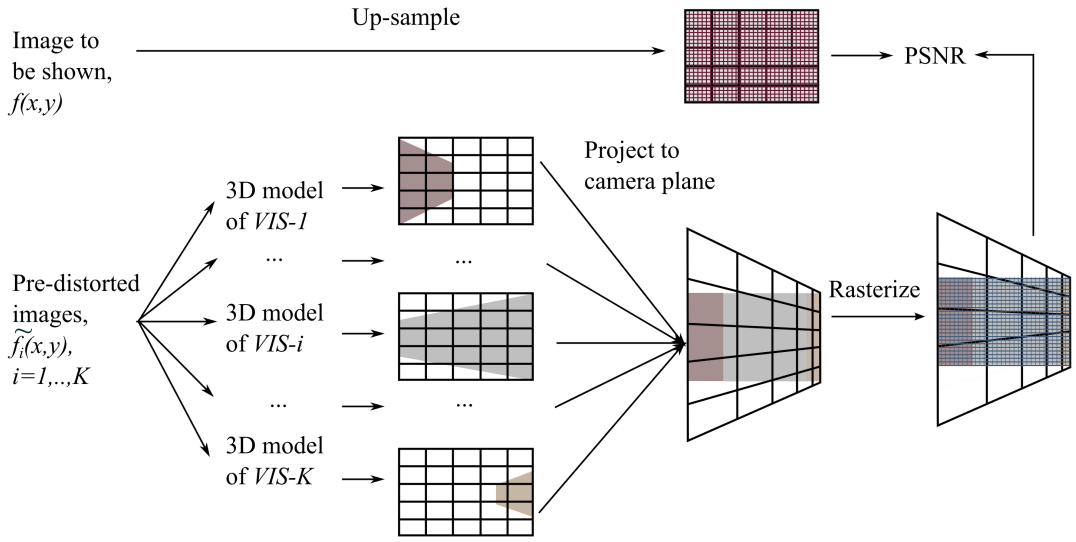

Figure 3.13 *Proposed image quality evaluation scheme.*

4. RESULTS AND ANALYSIS

The head-tracking system, and the motion-capture based system for visualization of virtual movie sets were implemented and tested. The the performance of the implementation was analysed. In this chapter we describe and analyze obtained implementations. Section 4.1 describes Kinect-based implementations: a head-tracked display, and Section 4.2 describes the motion-capture implementation and test results.

4.1 Head-tracked display

This section gives implementation results and analysis of the head-tracked display implemented with Kinect sensor and Matlab. A head-tracked display with Kinect v1 sensor was implemented in Matlab and Kinect for Windows SDK for skeleton tracking. System parameters are defined in Table 4.1. The model of the 3D scene

Table 4.1 *Systen parameters.*

System component	**Name**	**Specification**
Tracker	Kinect v1 sensor	Vertical FOV: 43 degrees
		Horizontal FOV: 57 degrees
		Frames per second: 30 FPS
Display	Dell E2016 20" LED IPS Monitor	Resolution: 1440 × 900 FPS
3D model	Set of cubes	Cube side length: 10 cm

was modelled by a set of three cubes with one side of 10 cm. The cubes were placed in front of and behind the monitor screen. Figure 4.1 shows pictures of the implemented setup in run. As can be seen from the pictures, a 2D image of the 3D objects is rendered from the viewpoint of the camera. A set of virtual 3D objects

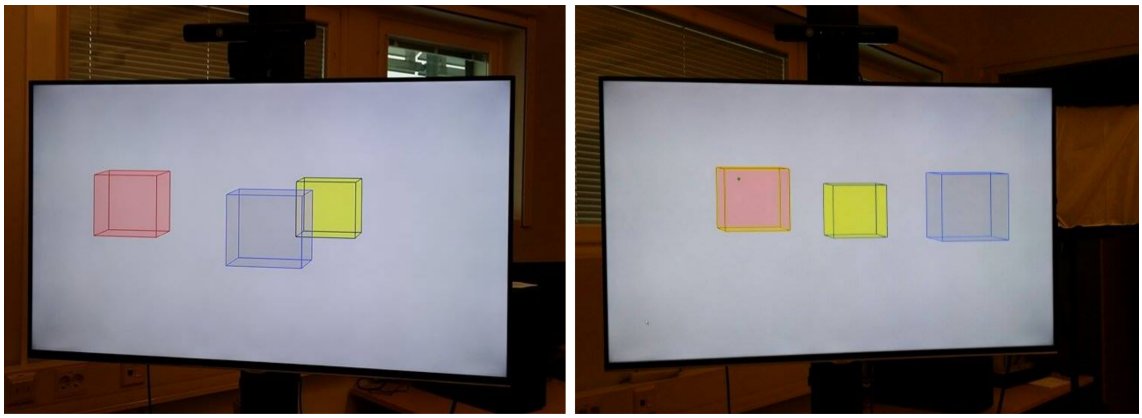

Figure 4.1 *Head-tracked display with Kinect v1 sensor. The rendering is done for the camera viewpoint.*

in the global 3D space that consisted of cubes, a pyramid and a sphere was used as a 3D model visualised on the display. Since Kinect sensor provides a noisy tracked data, a median filter was used to filter out the tracked outliers. The rendering part was done in the way explained in Section 3.1.

Analysis. Implemented head-tracked display with Kinect and Matlab performs the functionality of head-tracked spatially immersive displays. The head-tracked display demonstrates high processing frame rates. While the frame rate of Kinect for Windows SDK is 30 FPS, the frame rate of the system retains the rate 26-28 FPS. A virtual scene was modelled manually and consisted of cubes. Therefore, it is possible to represent a 3D virtual scene as a natural scene, without considerable work and implement visual analysis of the work of the system, which was done. A 3D model of the virtual scene was reconstructed as a natural scene, along with a wireframe model of the display. Then, 3D objects were placed in the precise positions of corresponding virtual objects in front and behind the modelled screen. Finally, a view of the natural scene from the precise position was compared to that of the scene displayed on the monitor. The visual testing showed correct behaviour of the display.

4.2 Projection-based CAVE-like system

An implementation of the projection-based immersive motion capture environment was done in the Matlab programming language. Two different testing set-ups were implemented. In this section the results and the analysis of the implemented systems

is given.

4.2.1 Performance evaluation of experimental set-up

The overall functionality of the implemented system is verified through a simplified set-up a front-projected visualization surface, an optical tracker and a camera put into the place of the tracked viewer. Technical specifications of the test are given in Table 4.2.

Table 4.2 *System parameters.*

System component	**Name**	**Specification**
Tracker	Optitrack V120:Trio	Vertical FOV: 43 degrees
		Horizontal FOV: 47 degrees
		Frames per second: 30, 60, 120 FPS
		Focal Length: 3.5 mm
		Imager Resolution: 640×480
Projector	Dell 1610HD	Resolution: 1920×1080
Source image	Test image of constant color	Resolution: 1920×1080
Camera	Nikon	Resolution: 6000×4000

Figure 4.2(a-b)) shows the simplified experimental set-up. Images taken by the tracked camera are analysed for their geometrical distortions. For reference, the geometry of the uncompensated images from the same camera position is captured. An example of one the test data images is shown in Figure 4.3. From the images captured from the test set-up, two metrics, rectangularity and the deviation of the corner angles are measured. The rectangularity, RECT, is measured by a minimum bounding rectangle method [59], where the rectangularity is the ratio between the area of the region and the area of the its minimum bounding rectangle (MBR). Let the image captured by the camera be $\tilde{f}'(x, y)$, $x = 0, \ldots, M-1$, $y = 0, \ldots, N-1$, with $M \times N$ being a resolution of the captured image. The camera and the projector are aligned in the way that the image captured by the camera covers at least all the projected imagery content. Let the area of the captured image $\tilde{f}'(x, y)$ that corresponds to the imagery content of the input image $f(x, y)$, that was geometry corrected to $\tilde{f}(x, y)$ and projected, be A pixels, and the area of the

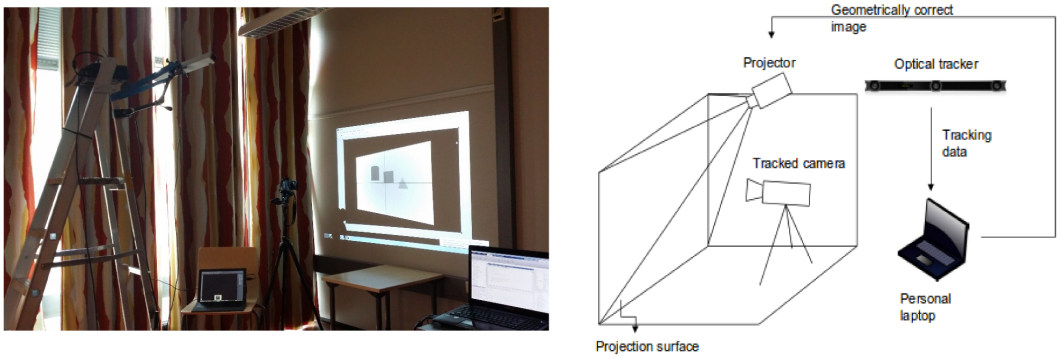

Figure 4.2 *A photo (a) and a diagram (b) of the experimental setup.*

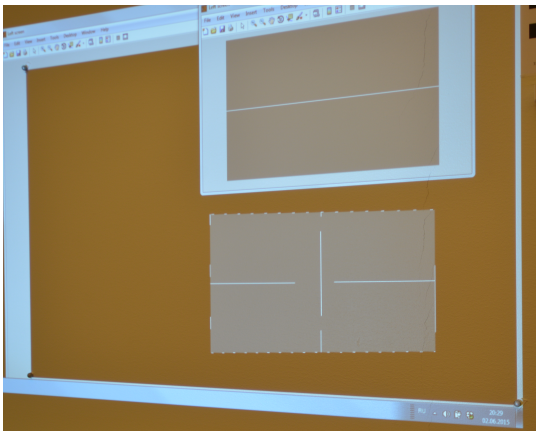

Figure 4.3 *Example of a test image, with an uncompensated image (top) and compensated (bottom) image for camera position.*

minimum bounding rectangle be B pixels. Then, the rectangularity is calculated as

$$\text{RECT} = \frac{A}{B}, \tag{4.1}$$

with $0 < \text{RECT} \leq 1$, and $\text{RECT} = 1$ meaning absolute rectangle. The deviation of the corner angles, α_i, $i = 1, \ldots, 4$ from optimal 90 degrees is measured by a mean absolute error, MAE_α, as

$$\text{MAE} = \frac{1}{4} \sum_{i=1}^{4} |\alpha_C - \alpha_i|, \tag{4.2}$$

where $\alpha_C = 90$ degrees, is a constant optimal angle, and α_i, $i = 1, \ldots, 4$ is a measured angle of the image $\tilde{f}'(x, y)$ captured with the tracked camera. The results

are shown in Figure 4.4. For reference, the same measures are presented for the

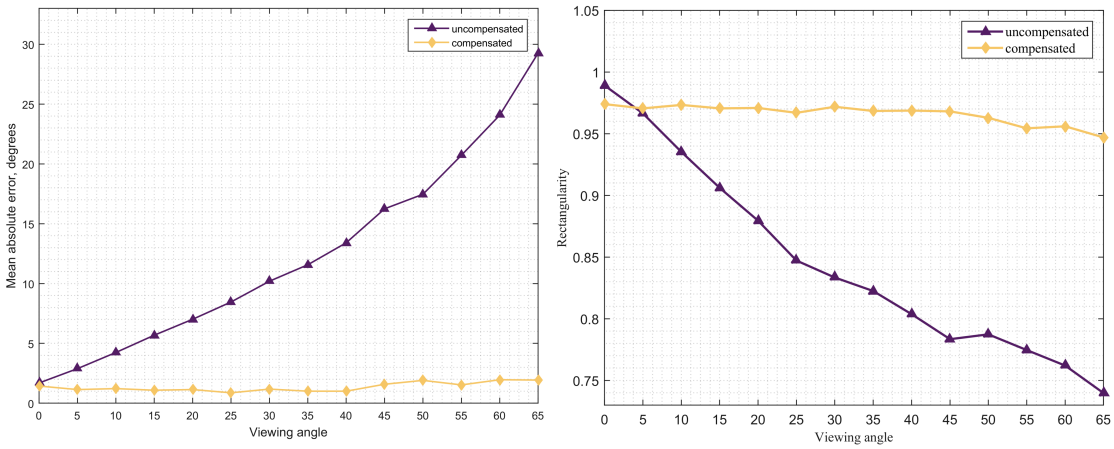

Figure 4.4 *Deviation from 90 degrees (left) and rectangularity (right) of the pre-distorted image (yellow) and the uncompensated (magenta) captured from the position of the virtual camera with different viewing angles.*

same image but without viewer position compensation. The captured image has a high rectangularity measure of around 0.95 and low angular deviation of 1-2 degrees even the extreme angles, which shows that the system is able to compensate for the different viewing angles.

4.2.2 Performance evaluation of simulated system

A simulated model produced a warped image for 6 sides of the room. Since the system is implemented in Matlab, a frame rate of 4 FPS was achieved. System main components' specifications are listed in Table 4.3. An order in which the warped im-

Table 4.3 *System parameters.*

System component	**Parameter**	**Specification**	
		Test 1	*Test 2*
Virtual camera	Vertical FOV	60 degrees	60 degrees
	Horizontal FOV	90 degrees	90 degrees
	Focal length	18 mm	18 mm
	Sensor size	36 × 24 mm	36 × 24 mm
Projectors	Resolution	1920 × 1080	640 × 360
Source image	Resolution	1920 × 1080	510 × 290

ages for each visualization surface are stored and represented in the thesis is shown in Figure 4.5. With the simulation framework in place, more comprehensive data and

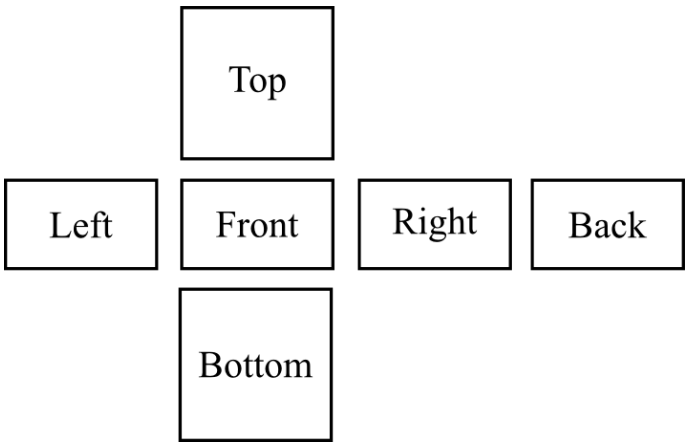

Figure 4.5 *Order of visualization surfaces in image containing all warped images.*

more detailed analysis of the effect of different parameters was conducted: rotation angle, image and display resolution, anti-aliasing filtering. Figure 4.6 shows images generated at different stages of the warping procedure and performance evaluation: the geometry corrected images for projectors (top), and the images formed during the mesh-based evaluation procedure - a mesh, formed by a composition of 6 meshes formed from the projected displays, projected back to the virtual camera plane and rasterized at high resolution (middle), and an upsampled reference image (bottom). A lower image is a pre-distorted for a 85 degrees viewing angle image, represented with mesh and projected to the camera plane. A considerable amount of blurring in the center of the image appears due to a high slant of the viewing angle, which causes drastic sample rate convention and high amount of low-pass filtering to be performed. Figures 4.7 and 4.8 show warped image for viewing angles of 0, 45, and 85 degrees along Y axis. The following regular-to-irregular sampling schemes and filtering were tested:

- Low-pass pre-filtering with Kaiser window and nearest/(bi)linear/ (bi)cubic and cubic spline interpolation.
- Low-pass pre-filtering with Butterworth filter and nearest/(bi)linear/ (bi)cubic and cubic spline interpolation.
- Low-pass pre-filtering with Gaussian filter and nearest/(bi)linear/ (bi)cubic and cubic spline interpolation.

- Low-pass filtering and re-sampling as a least-squares spline fit.

Comparison of interpolation schemes for different projector-camera resolutions. A comparison between several conventional regular-to-irregular sampling schemes for rendering the warped image is presented in Figure 4.9 using the proposed mesh based quality evaluation. Intuitively the most influential parameter being the angle between the viewing direction and the display surface, the PSRN results are shown for this case. Nearest neighbour, linear, cubic, spline and least-squares spline interpolation are compared together with an application of an adaptive anti-aliasing filter. The results show that the least-squares spline provides better results. Also, an effect of image and display resolutions was tested. The PSNR for the following cases was compared:

- Image resolution > projector resolution.
- Image resolution < projector resolution.
- Image resolution = projector resolution.

A useful finding is that for the case when image resolution < projector resolution. For this case, the best interpolation scheme for all anti-aliasing filters was nearest neighbour. The results for the case when an image of the resolution of 510×290 is geometry corrected, and projectors with resolution of 1920×1080 is used for visualization, are shown in Figure 4.10. For this case, the best interpolation scheme for almost all anti-aliasing filters is nearest neighbour. The reason of the result shown at Figure 4.10 is that the projector resolution is much higher, which is sufficient to provide the best approximation of the source image. For the cases when image resolution = projector resolution, the best interpolation technique is provided by the proposed leas-squares interpolation method, based on splines (Figure 4.9). For the cases when image resolution > projector resolution, all the interpolation schemes showed a fair result. For example, in the result shown in Figure 4.11, projectors with the resolution of 1920×1080 were used with the 640×360 reference image. In this case, the projector resolution is much smaller, and the reference image cannot be approximated close enough by any of the interpolation schemes. Therefore, as can be seen from the Figure 4.11, all the interpolation schemes produce similar level of result.

Comparison of the filters. A comparison of the filters has also been conducted. A total of three different window filters were used in the test data: Kaiser, Gaussian and Butterworth filter. Different filter parameters were tested and relative filter performance was analysed. The filters with parameters used for the tests are presented in Table 4.4. Filters with sharper transition bands, such as Kaiser filter, produce a

Table 4.4 *Filter specifications.*

Filter type	**Parameters**	
	Test 1	*Test 2*
Butterworth	$n = 1$	$n = 3$
Gaussian	$n = 71$, $\alpha = 3.5$	$n = 21$, $\alpha = 2.5$
Kaiser	$n = 71$, $\beta = 7$	$n = 51$, $\beta = 5$

better attenuation. Such filters as Butterworth and Gaussian have smoother transition regions, which leave more spectral components after filtering, which can still appear as aliasing artefacts in the visualised image.

Variation of rotation angles. The test for simple rotations around Y axis, Z axis were carried out along with tests for more complex rotations, along Y,Z and X,Y,Z axes. The results showed no big variation of the PSNR for complex and simple rotations. The difference that can be pointed out is that even simple interpolation techniques show good approximations in the case of simple rotations, while for more complex rotations bicubic and spline interpolations provide better results.

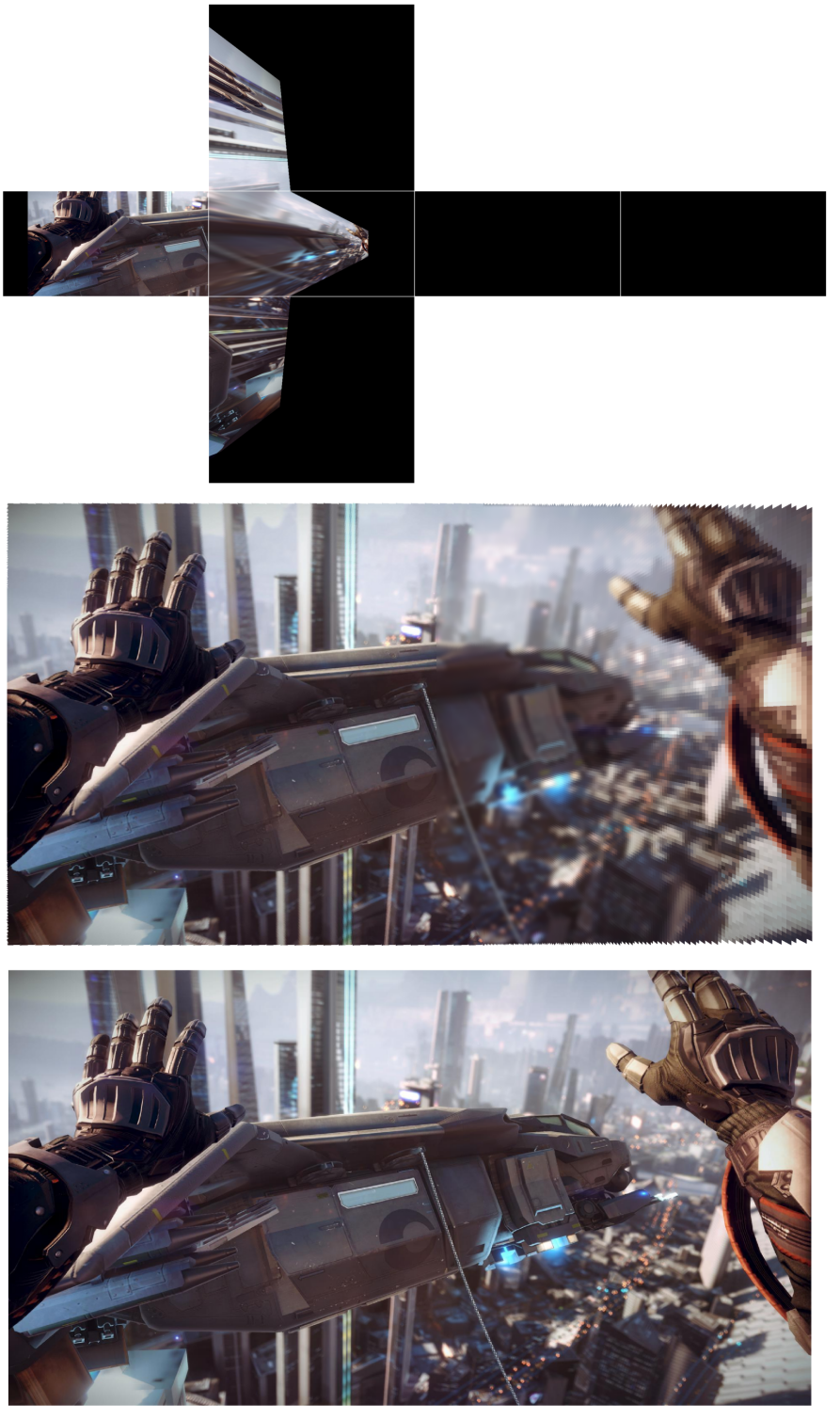

Figure 4.6 *From top to down: an image of warped sides of the cube, a mesh of the warped images of the cube projected back to the camera plane and rasterized at high resolution, and an upsampled reference image.*

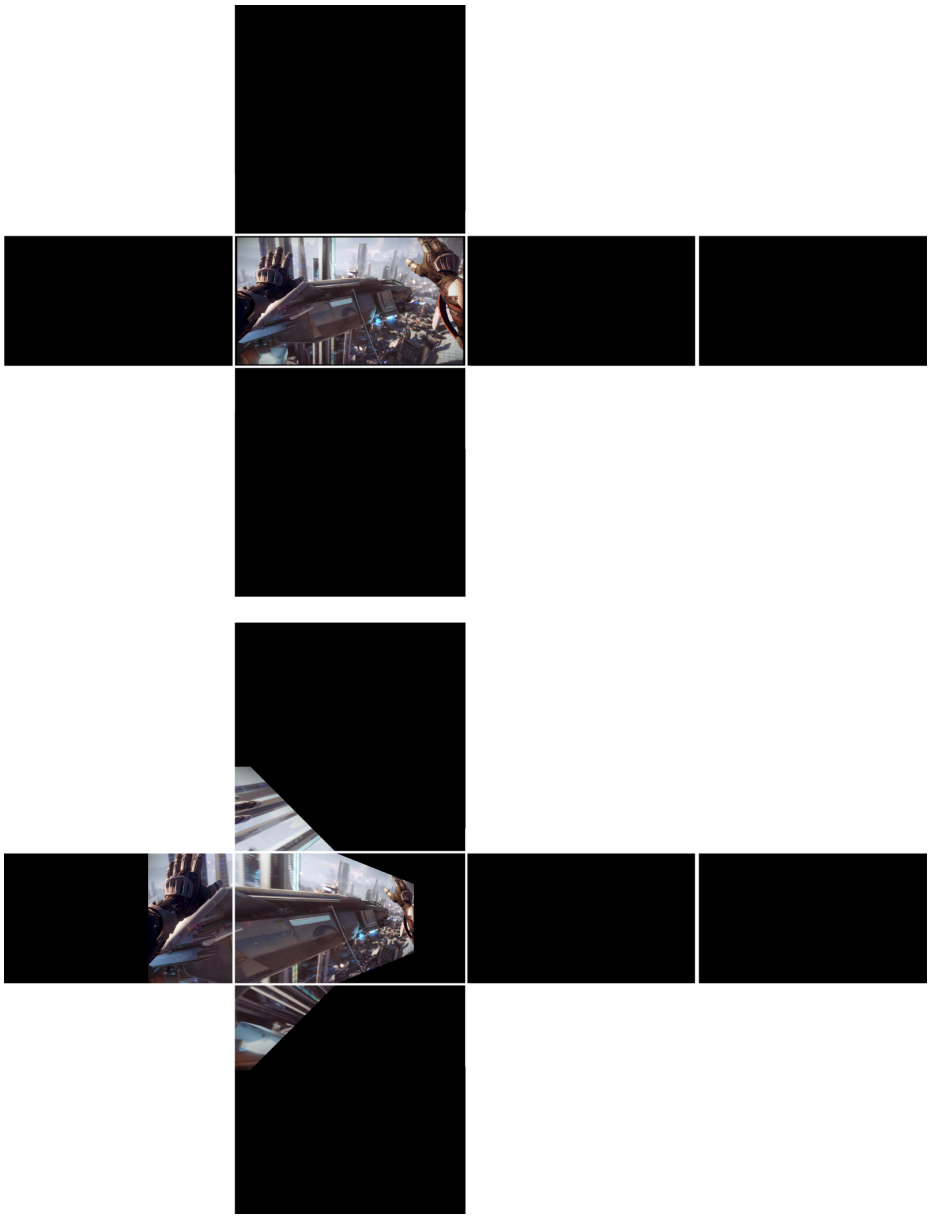

Figure 4.7 *Different viewing angles along Y axis. From top to bottom: 0, 45 degrees, respectively.*

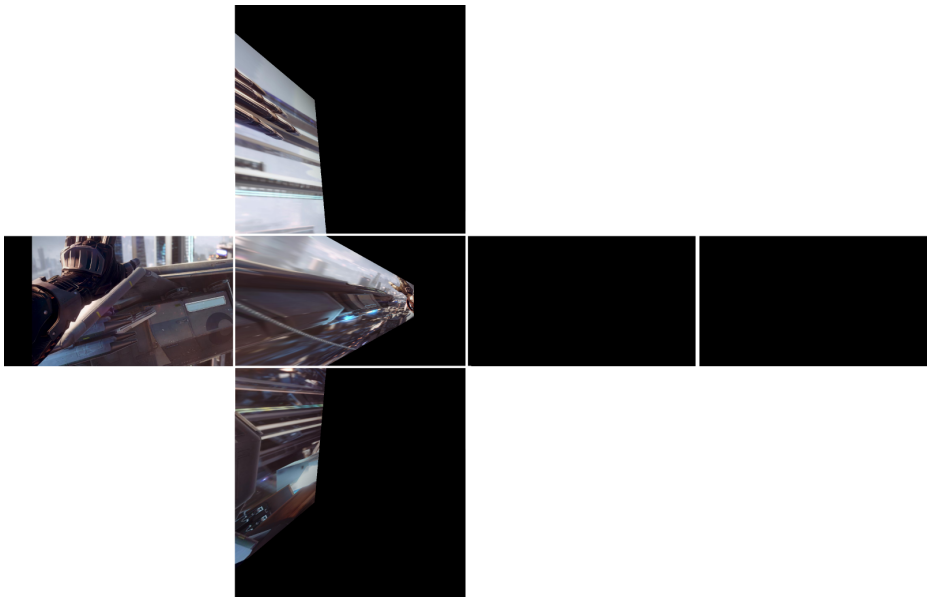

Figure 4.8 *Viewing angles along Y axis: 85 degrees.*

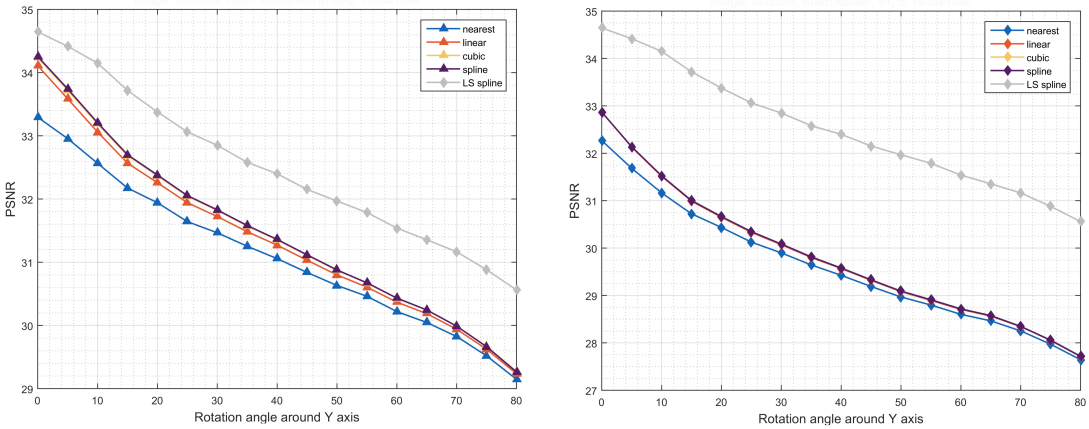

Figure 4.9 *Comparison of the quality produced by different interpolation techniques with anti-aliasing done with Butterworth (left), Kaiser (right) filters for the image of resolution* 1920×1080 *and projector resolutions* 1920×1080 *and least-squares splines (grey line).*

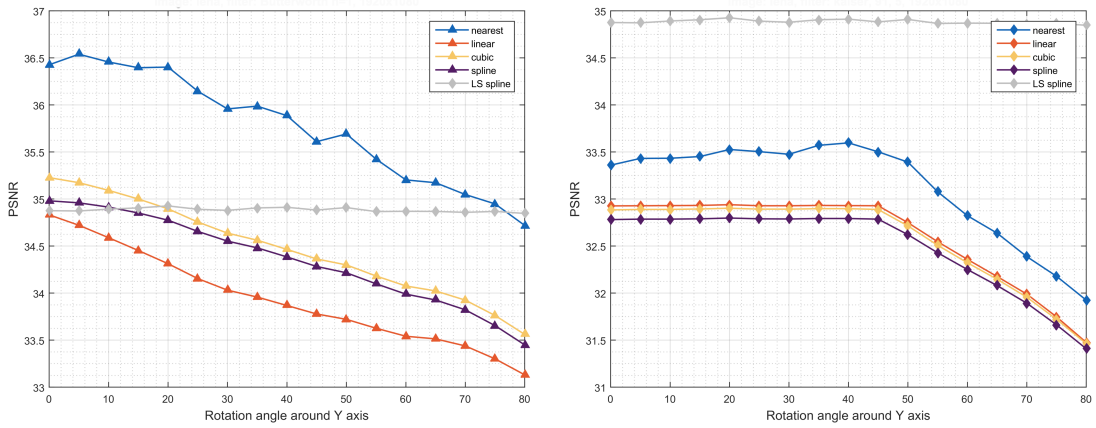

Figure 4.10 *Comparison of the quality produced by different re-sampling techniques for the (lena) image of resolution* 510×290 *and projector resolutions* 1920×1080 *with Butterworth(left) and Kaiser (right) filter.*

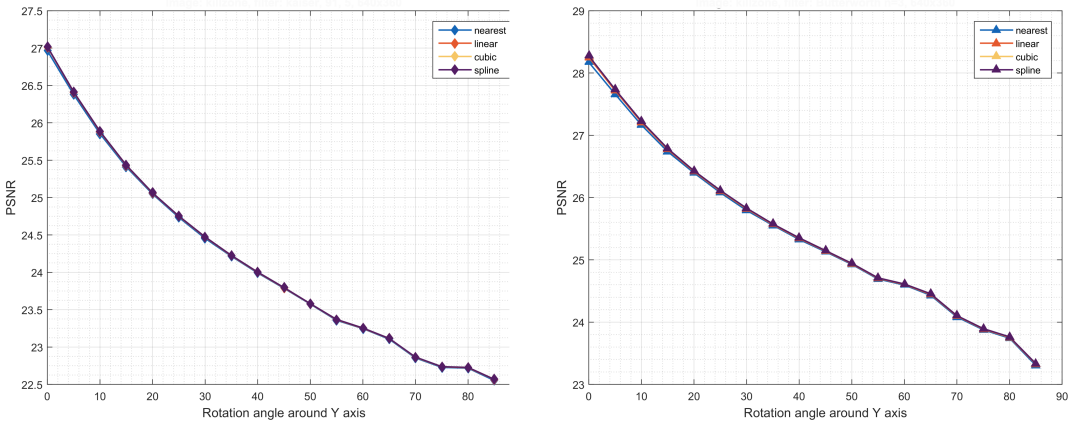

Figure 4.11 *Comparison of the quality produced by different re-sampling techniques for the image of resolution* 1920×1080 *and projector resolutions* 640×360 *with Kaiser (left) and Butterworth (right) filters.*

5. CONCLUSIONS

Currently it is possible to provide good visualization to other crew members in a motion capture environment except the actor. The goal of the presented work was to develop a tool for providing also the actor with proper visualization of the virtual content without confining him/her with additional peripherals like VR glasses. The taken approach has similarities with CAVE-like virtual reality systems, however, it utilizes an image processing approach.

A geometry correction procedure for visualisation of images on large-scale projection surfaces without distortions was developed. The solution included adaptive filtering for regular-to-irregular re-sampling. Also, a more efficient way to implement re-sampling and filtering with the help of spline basis functions was introduced. A simulation model of the CAVE-like projection based system and a simplified experimental set-up were developed. The simulated model along with performance evaluation framework allow for testing the solution with different parameters without actual usage of projectors, which greatly improves testing time. With the help of these, the entire set of test images was derived and the effect of different parameters on the PSNR was analysed. The tests included distinct rotations around each of the axis, along with complex rotations around several axes at a time. Also, different interpolation schemes and different filters were tested. The differences in image and projector resolutions showed very interesting results. Moreover, the tests provide solutions for parameter adjustments for better performance of the proposed system. Moreover, the proposed mesh based quality evaluation solves an important alignment issue with computing objective quality metrics, and may find other uses in other applications where e.g. the quality of spatially immersive displays is evaluated. The relatively simple model can also be extended with additional effects such as nonuniform intensity of pixels or non-Lambertian display surfaces to achieve greater realism.

Future steps in this area include integration of several projectors in the actual system

set-up, implementation of the system in real-time with the help of programming on Graphical Processing Units (GPUs) and actual linking the system with the image rendering engine. Also, an extension of the system to the more natural case, i.e. when the visualization surfaces are non-planar, can be done in future. In this case, a 3D geometry model of the visualization surfaces can be achieved with the help of structured light, depth-form stereo or time-of-flight techniques. Also, the adjustment of the radiometric compensations for the color content of the image can be added to compensate for imperfections of the visualization surfaces [42].

BIBLIOGRAPHY

[1] A. Aldroubi, H. Feichtinger, Complete iterative reconstruction algorithms for irregularly sampled data in spline-like spaces, In Proceedings of ICASSP, Vol. 3, 1997, pp. 1857-1860.

[2] W. Bares, S. McDermott, C. Boudreaux, S. Thainimit, Virtual 3D camera composition from frame constraints, In Proceedings of MM, October, 2000, ACM, pp. 177-186.

[3] O. Bimber, R. Raskar, Spatial augmented reality: merging real and virtual worlds, CRC Press, 2005.

[4] O. Bimber, G. Wetzstein, A. Emmerling, C. Nitschke, Enabling View-Dependent Stereoscopic Projection in Real Environments, In Proceedings of ISMAR, 2005, IEEE, pp. 14-23.

[5] J. Biswas, M. Veloso, Depth Camera Based Indoor Mobile Robot Localization and Navigation, In Proceedings of ICRA, May, 2012, IEEE, pp. 1697-1702.

[6] M. Brown, A. Majumder, R. Yang, Camera-based calibration techniques for seamless multiprojector displays, IEEE Transactions on Visualization and Computer Graphics, Vol. 11, No. 2, 2005, pp. 193-206.

[7] C. Chen, Y. Hung, C. Chiang, J.Wu, Range data acquisition using color structured lighting and stereo vision, Image and Vision Computing, 1997, pp. 445-456.

[8] A. Comport, E. Marchand, M. Pressigout, F. Chaumette, Real-time markerless tracking for augmented reality: the virtual visual servoing framework, IEEE Transactions on Visualization and Computer Graphics, Vol. 12, No. 4, 2006, pp. 615-628.

[9] C. Cruz-Neira, D.J. Sandin, T.A. DeFanti, Surround-screen projection-based virtual reality: the design and implementation of the CAVE, In Proceedings of SIGGRAPH, Anaheim, CA, USA, August 02-06, 1993, ACM New York, NY, pp. 135-142.

[10] J. Cunha, E. Pedrosa, C. Cruz, A. J. R. Neves, N. Lau, Using a Depth Camera for Indoor Robot Localization and Navigation, DETI/IEETA-University of Aveiro, Portugal, 2011.

[11] H. Davson, Physiology of the Eye, Elsevier, 2012.

[12] C. De Boor, A practical guide to splines, Mathematics of Computation, 1978.

[13] C. De Boor, Spline toolbox for use with MATLAB: user's guide, Version 3, MathWorks, 2005.

[14] N. Dodgson, N, Quadratic interpolation for image resampling, IEEE Transactions on Image Processing, 1997, pp. 1322-1326.

[15] H. Feichtinger, K. Gröchenig, Theory and practice of irregular sampling, in Wavelets: Mathematics and Applications, pp. 305-363, CRC Press, 1994.

[16] F. Garcia, B. Mirbach, R. Orsello, T. Solignac, 3d time-of-flight camera system and position/orientation calibration method therefor, U.S. Patent Application No. 13/058,962.

[17] R. C. Gonzalez, R. E. Woods, and S. L. Eddins, Digital Image Processing Using MATLAB, 2nd ed., Gatesmark Publishing, 2009, 827 p.

[18] O. Grau, T. Pullen, G. Thomas, A combined studio production system for 3-D capturing of live action and immersive actor feedback, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 14, Issue 3, 2004, pp. 370-380.

[19] M. Gross, S. Würmlin, M. Naef, E. Lamboray, C. Spagno, A. Kunz, K. Strehlke, blue-c: a spatially immersive display and 3D video portal for telepresence, ACM Transactions on Graphics, Vol. 22, No. 3, 2003, pp. 819-827

[20] M. D. Grossberg, H. Peri, S. K. Nayar, P. N. Belhumeur, Making one object look like another: Controlling appearance using a projector-camera system, In Proceedings of CVPR, Vol. 1, 2004, IEEE, pp. I-452. CVPRs

[21] J. Gühring, Dense 3-d surface acquisition by structured light using off-the-shelf components, in Photonics West 2001-Electronic Imaging, 2001, International Society for Optics and Photonics, pp. 220-231.

[22] O. Hall-Holt, S. Rusinkiewicz, Stripe boundary codes for real-time structured-light range scanning of moving objects, In Proceedings of ICCV, 2001, pp. 359-366.

[23] R. Hartley, A. Zisserman, Multiple view geometry in computer vision, Cambridge university press, 2003.

[24] M. J. Hogan, J. E. JA Weddell, Histology of the human eye: an atlas and textbook, 1971.

[25] I. Ishii, A Coded Structured Light Projection Method for High-Frame-Rate 3D Image Acquisition, INTECH Open Access Publisher, 2012. Available(accessed 23.10.2015): http://www.intechopen.com/books/advanced-image-acquisition-processing- techniques-and-applications-i/a-coded-structured-light-projection-method- for-high-frame-rate-3d-image-acquisition

[26] M. Jalobeanu, G. Shirakyan, G. Parent, H. Kikkeri, B. Peasley, A. Feniello, Reliable Kinect-based Navigation in Large Indoor Environments, In Proceedings of ICRA, Seattle, WA, USA, 2015, IEEE, pp. 495-502.

[27] V. Jaswa, CAVEvis: distributed real-time visualization of time-varying scalar and vector fields using the CAVE virtual reality theater, In Proceedings of VIS97, October, 1997, IEEE Computer Society Press Los Alamitos, CA, USA, pp. 301-308.

[28] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, A. Kolb, Real-time 3D reconstruction in dynamic scenes using point-based fusion, In Proceedings of 3DV, June, 2013, pp. 1-8.

[29] F. Kensaku, M. D. Grossberg, S. K. Nayar, A projector-camera system with real-time photometric adaptation for dynamic environments, Computer Vision and Pattern Recognition, Vol. 1, 2005.

[30] R. G. Keys, Cubic Convolution Interpolation for Digital Image Processing, IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. 29, No. 6, 1981, pp. 1153-1160.

[31] R. Kooima, Generalized perspective projection, School of Elect. Eng. and Computer Science, 2008, pp. 1-7.

[32] K. Li, Y. Chen, Optical blending for multi-projector display wall system, In Proceedings of LEOS, 1999, IEEE, Vol. 1, pp. 281-282.

[33] B. Y. Li, A. Mian, W. Liu, A. Krishna, Using Kinect for face recognition under varying poses, expressions, illumination and disguise, In Proceedings of WACV, January, 2013, IEEE, pp. 186-192.

[34] A. Majumder, Z. He, H. Towles, G. Welch, Color calibration of projectors for large tiled displays, In Proceedings of Visualization, October 8-13, 2000, IEEE, p. 102.

[35] L. McMillan, G. Bishop, Head-tracked stereoscopic display using image warping, In Symposium on Electronic Imaging: Science and Technology, 1995, International Society for Optics and Photonics, pp. 21-30.

[36] A. Menache, Understanding Motion Capture for Computer Animation, 2nd. Edition, Morgan Kaufmann Publishers Inc., San Francisco, CA, 2010.

[37] S. K. Mitra, Digital Signal Processing: A Computer-Based Approach, 2nd ed., McGraw-Hill, 2010.

[38] T. P. Monks, J. N. Carter, C. H. Shadle, Colour-encoded structured light for digitisation of real-time 3D data, In Proceedings of the International Conference on Image Processing and its Applications, Netherlands, April 7-9, 1992, IET, pp. 327-330.

[39] H. Morita, K. Yajima, S. Sakata, Reconstruction of surfaces of 3-d objects by m-array pattern projection method, In Proceedings of ICCV, 1988, pp. 468-473.

[40] H. H. Mousavi, M. Khademi, A Review on Technical and Clinical Impact of Microsoft Kinect on Physical Therapy and Rehabilitation, Journal of Medical Engineering, 2014, p. 16.

[41] S. K. Nayar, M. Gupta, Diffuse structured light, In Proceedings of ICCP, April, 2012, pp. 1-11.

[42] S. K. Nayar, H. Peri, M. D. Grossberg, P. N. Belhumeur, A projection system with radiometric compensation for screen imperfections, In ICCV workshop on PROCAMS, Vol. 3, 2003.

[43] A. Oliver, S. Kang, B. C. Wunsche, B. MacDonald, Using the Kinect As a Navigation Sensor for Mobile Robotics, In Proceedings of IVCNZ, Dunedin, New Zealand, November 26-28, 2012, ACM New York, NY, pp. 509-514.

[44] J. Pagés, J. Salvi, Coded light projection techniques for 3D reconstruction, J3eA, 4(HORS SÉRIE 3), 001, 2005.

[45] J. A. Parker, R. V. Kenyon, D. E. Troxel, Comparison of interpolating methods for image resampling, IEEE Transactions on Medical Imaging, Vol. MI-2, No. 1, 1983, pp. 31-39.

[46] T. W. Parks, C. S. Burrus, Digital filter design, Wiley-Interscience, 1987.

[47] J. L. Potsdamer, M. D. Altschuler, Surface measurement by space-encoded projected beam systems, Computer graphics and image processing, Vol. 18, No. 1, 1982, pp. 1-17.

[48] R. Raskar, Immersive planar display using roughly aligned projectors, In Proceedings of VR, New Brunswick, NJ, 2000, IEEE, pp. 109-115.

[49] R. Raskar, M. S. Brown, R. Yang, W. C. Chen, G. Welch, H. Towles, ... and H. Fuchs, Multi-projector displays using camera-based registration, In Proceedings of VIS, 1999, IEEE Computer Society Washington, DC, pp. 161-522.

[50] R. Raskar, J. Van Baar, P. Beardsley, T. Willwacher, S. Rao, C. Forlines, iLamps: Geometrically Aware and Self-Configuring Projectors, In ACM SIGGRAPH Courses, 2006, ACM, pp. 809-818.

[51] R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, H. Fuchs, The Office of the Future: A Unified Approach to Image-Based Modeling and Spatially Immersive Displays, In Proceedings of SIGGRAPH, 1998, ACM Press, pp. 179-188.

[52] J. Rolland, Head-mounted display systems, in Encyclopedia of Optical Engineering, New York, NY : Marcel Dekker, 2005, p. 1.

[53] H. Sankaran, M. Georgiev, A. Gotchev, K. Egiazarian, Non-uniform to uniform image resampling utilizing a 2D farrow structure, In Proceedings of SMMSP, 2007.

[54] D. Scharstein, R. Szeliski, High-accuracy stereo depth maps using structured light, In Proceedings of CVPR, June, 2003, IEEE, Vol. 1, pp. I-195.

[55] C. E. Shannon, Communication in the presence of noise, In Proceedings of IRE, Vol. 37, No. 1, 1949, pp. 10-21.

[56] R. Szeliski, Computer vision: algorithms and applications, Springer Science and Business Media, 2010.

[57] J. Tauscher, W. O. Davis, D. Brown, M. Ellis, Y. Ma, M. E. Sherwood, D. Bowman, M. P. Helsel, S. Lee, J. W. Coy, Evolution of MEMS scanning mirrors for laser projection in compact consumer electronics, in MOEMS-MEMS, International Society for Optics and Photonics, 2010, pp. 75940A-75940A.

[58] P. Thévenaz, T. Blu, M. Unser, Image interpolation and resampling, Handbook of medical imaging, processing and analysis, 2000, pp. 393-420.

[59] G. T. Toussaint, Solving geometric problems with the rotating calipers, In Proceedings of MELECON, Athens, Greece, May 24-26, 1983, Vol. 83, p. A10.

[60] J. Van Baar, T. Willwacher, S. Rao, R. Raskar, Seamless multi-projector display on curved screens, In the workshop on Virtual environments, May, 2003, ACM Press, pp. 281-286.

[61] L. Vera, J. Gimeno, I. Coma, and M. Fernández, Augmented mirror: interactive augmented reality system based on Kinect, In Human-Computer Interaction (INTERACT), Springer Berlin Heidelberg, 2011, IEEE, pp. 483-486.

[62] Kinect for Windows SDK. Available(accessed on 13.05.2015): http://msdn.microsoft.com

[63] Unity technologies. Available(accessed on 10.10.2015): https://unity3d.com/company

[64] Unreal engine 4. Available(accessed on 10.10.2015): https://www.epicgames.com/

[65] Motion Builder overview. Available(accessed on 10.10.2015): http://www.autodesk.com/products/motionbuilder/overview

[66] Blender. Available(accessed on 10.10.2015): https://www.blender.org/